

FHWS

# iOS Programmierung (mit Swift)

Peter Braun, Florian Bachmann & Andreas Wittmann  
[@pe\\_braun](https://twitter.com/pe_braun)   [@florianbachmann](https://twitter.com/florianbachmann)   [@anwittmann](https://twitter.com/anwittmann)

Deutsche Telekom AG  
FHWS - Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt  
#FHWSSwift

# Agenda

1. **Introduction** – Organisatorisches
2. **First iOS-Project** – Hello World, **First iOS-Project** – Still Hello World (now with Code 😊)
3. **Swift**, Wait!, What about Objective-C?, Why Swift?
4. **A (not so) Quick Tour**
5. **Documentation**
6. **The basics** – iOS Architecture & more
7. **User Interfaces** – View Controller, Autolayout & Size Classes
8. **Storyboard & Segues**
9. **Tables & NavigationController**
10. **TabBarController**
11. **Notifications**
12. **Picker Views**
13. **Touches, Gestures, 3D Touch, Peek & Pop**
14. **StackViews**
15. **Networking** – JSON & Dependency Managers
16. **WebKit**
17. **Maps**
18. **Storage & Data persistency** – NSUserDefaults, NSKeyedArchiver & Core Data
19. **ObjC**

# The Basics

## iOS Architecture & more

# Platform Components - Tools



Xcode



Betas FTW! :)



iOS Simulator



Instruments



AppCode



Reveal



Spark



Dash



PaintCode

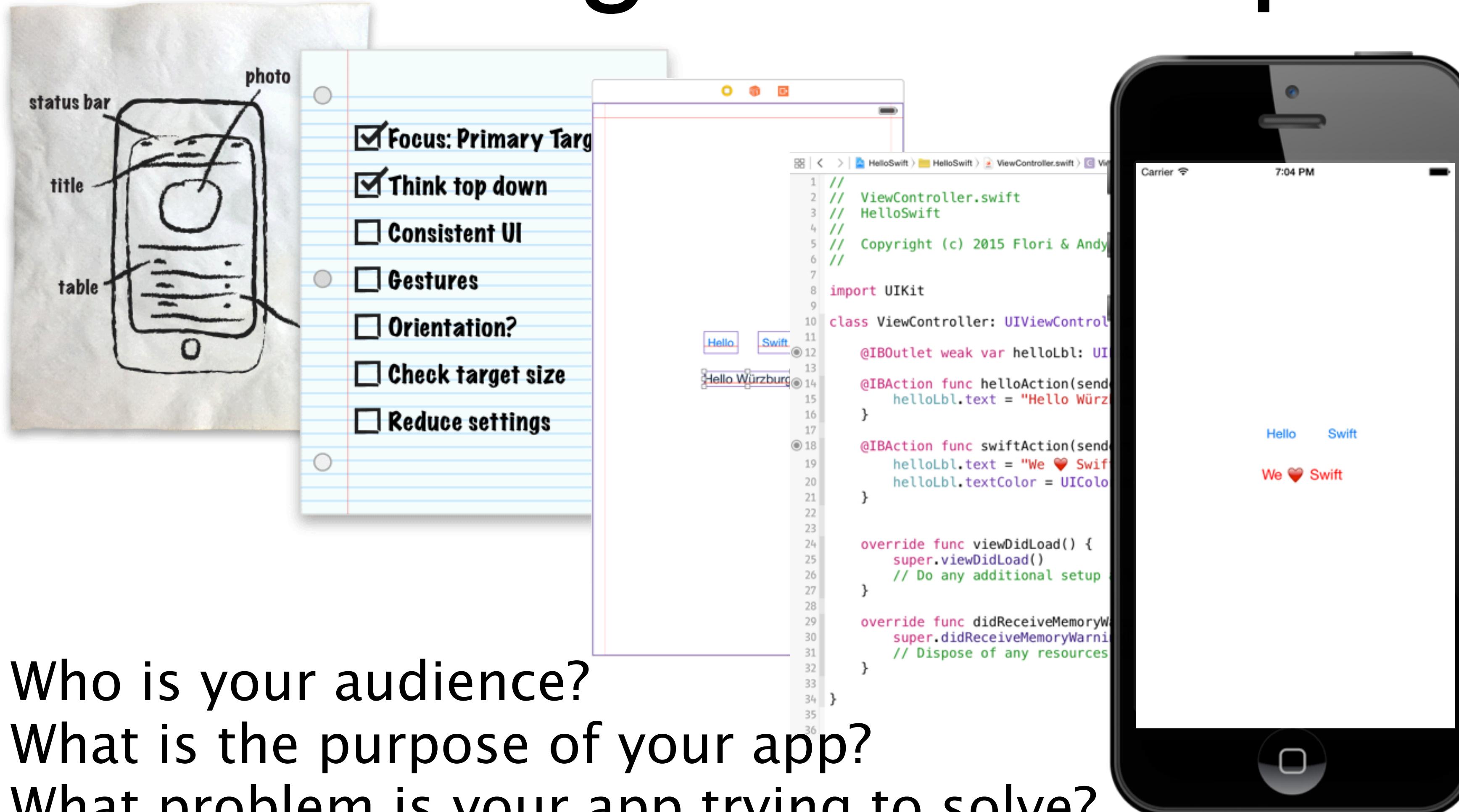


SimPholders



Alcatraz

# Defining the concept



1. Who is your audience?
2. What is the purpose of your app?
3. What problem is your app trying to solve?
4. What content will your app incorporate?

<https://developer.apple.com/videos/wwdc/2014/#223>

# Golden Rules of Interface Design

## The eight by Ben Shneiderman

1. Strive for consistency.
2. Enable frequent users to use shortcuts.
3. Offer informative feedback.
4. Design dialog to yield closure.
5. Prevent errors
6. Permit easy reversal of actions.
7. Support internal locus of control.
8. Reduce short-term memory load.

# Golden Rules of Interface Design

Andy & Flori

1. Keep it simple! It is an App!
2. Our mothers need to understand it
3. Screens are small, fingers are thick
4. Focus on the key feature
5. one App per use case!
6. Gestures are still new, explain them
7. make it conform to the HIG

# HIG - Human Interface Guidelines

UI Design Basics

## Designing for iOS

iOS App Anatomy

Adaptivity and Layout

Starting and Stopping

Navigation

Modal Contexts

Interactivity and Feedback

Animation

Branding

Color and Typography

Icons and Graphics

Terminology and Wording

Integrating with iOS

Design Strategies

iOS Technologies

UI Elements

Icon and Image Design

On This Page ▾

## Designing for iOS

iOS embodies the following themes:

- **Deference.** The UI helps people understand and interact with the content, but never competes with it.
- **Clarity.** Text is legible at every size, icons are precise and lucid, adornments are subtle and appropriate, and a sharpened focus on functionality motivates the design.
- **Depth.** Visual layers and realistic motion impart vitality and heighten people's delight and understanding.



HIG

# Mobile - Focus

- Memory is limited
- Only one App at a time
- It is a phone, not an iMac (small)
- keep restrictions in mind
- Only one app at a time
- Context is key (do one thing and do it well)
- Data is stored per app
- Data exchange between apps is hard
- minimized Text input (despite developing a Text editor)
- information flow from top to bottom

# iOS Res

A quick reference for iOS devices.

# iOS Screen Sizes

								
FILTER	iPhone 5	iPhone 6	iPhone 6 Plus	iPad Mini	iPad Air	iPad Pro	Apple Watch 38mm	Apple Watch 42mm
RESOLUTION	640 x 1136	750 x 1334	1242 x 2208 <sup>①</sup>	768 x 1024	1536 x 2048	2048 x 2732	272 x 340	312 x 390
LOGICAL RESOLUTION	320 x 568	375 x 667	414 x 736	768 x 1024	768 x 1024	<u>1024 x 1366?</u> <sup>①</sup>		
ASSOCIATED DEVICES	iPhone 5, 5C, 5S, iPod Touch 5g	iPhone 6, iPhone 6s	iPhone 6 Plus, iPhone 6s Plus	iPad Mini 2, iPad Mini 3, iPad Mini 4	iPad 3, iPad 4, iPad Air, iPad Air 2			
PPI	326 ppi file@3x.png	326 ppi	401 ppi	264 ppi file@2x.png	264 ppi	326 ppi	326 ppi	
ASPECT RATIO	9:16	9:16	9:16	3:4	3:4	3:4	4:5	4:5

# App Development

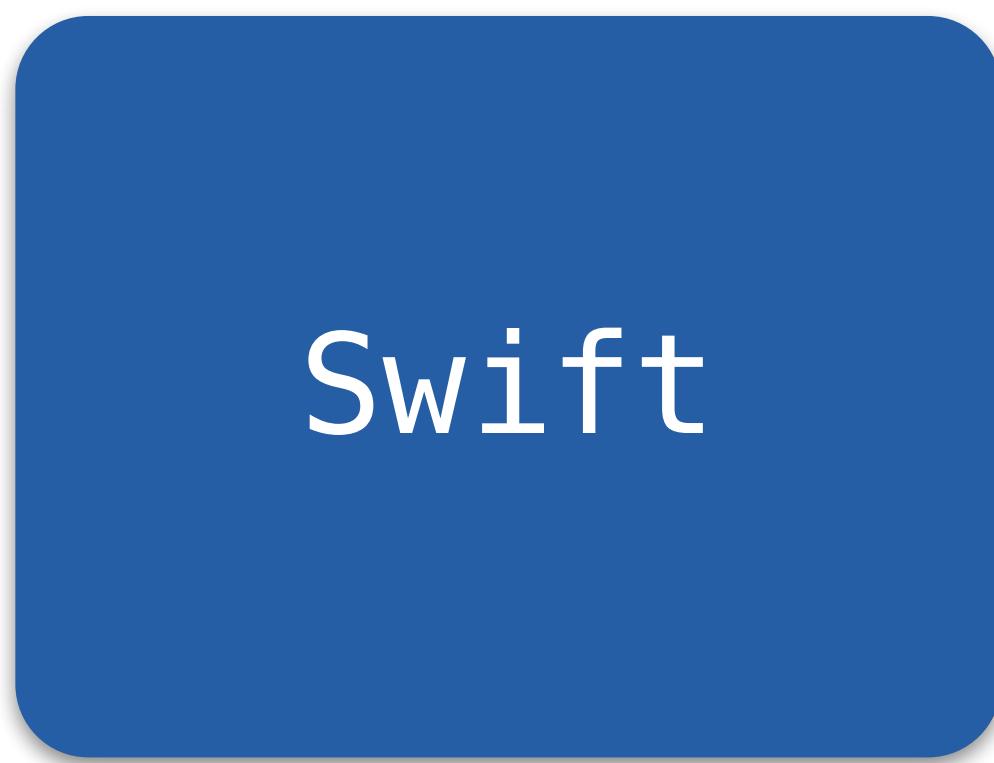
Tools by Apple

Xcode

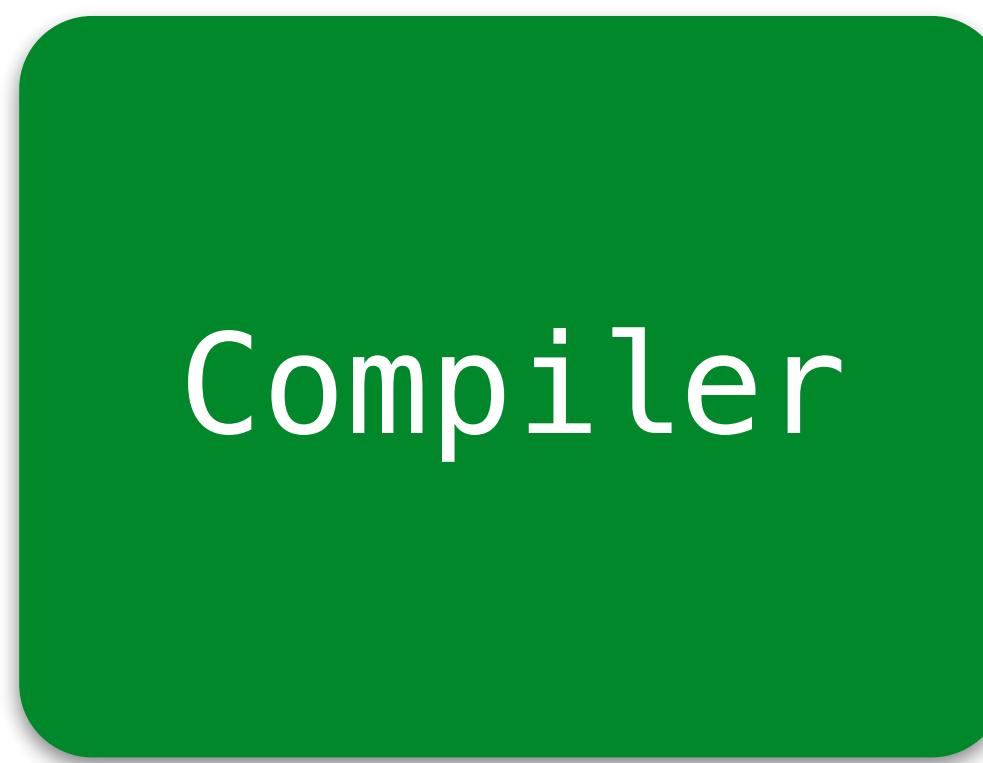
Instruments

iOS  
Simulator

# App Compiling Steps



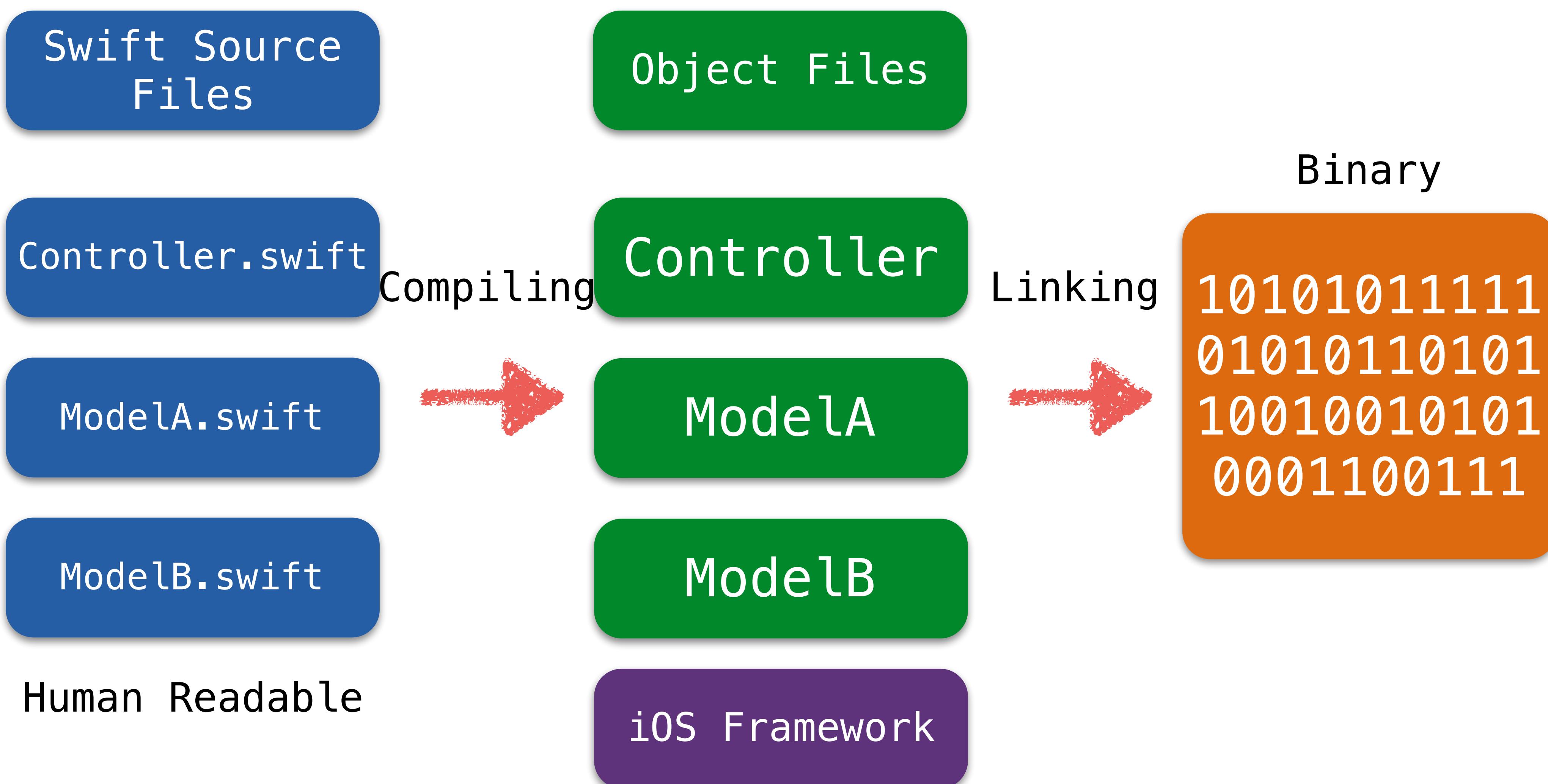
Human Readable



10101011111  
01010110101  
10010010101  
0001100111

machine code

# App Compiling Steps



# App Ingredients

Swift

MVC

Xcode

Frameworks

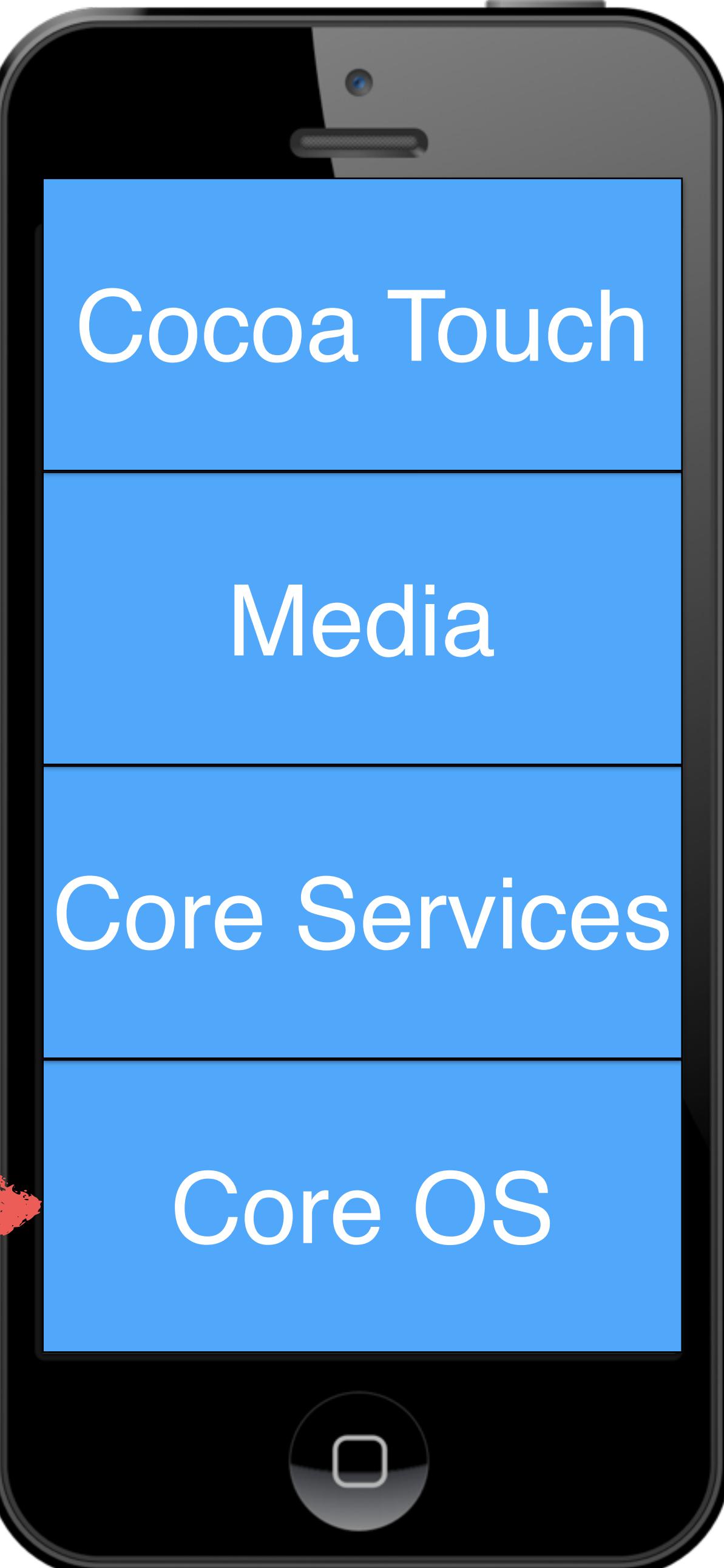
Foundation

UIKit

Instruments

Compiler

iOS  
Simulator



# What is iOS

## 1. Core OS Layer

OSX Kernel

Power Management

Mach 3.0

Keychain Access

BSD

Certificates

Sockets

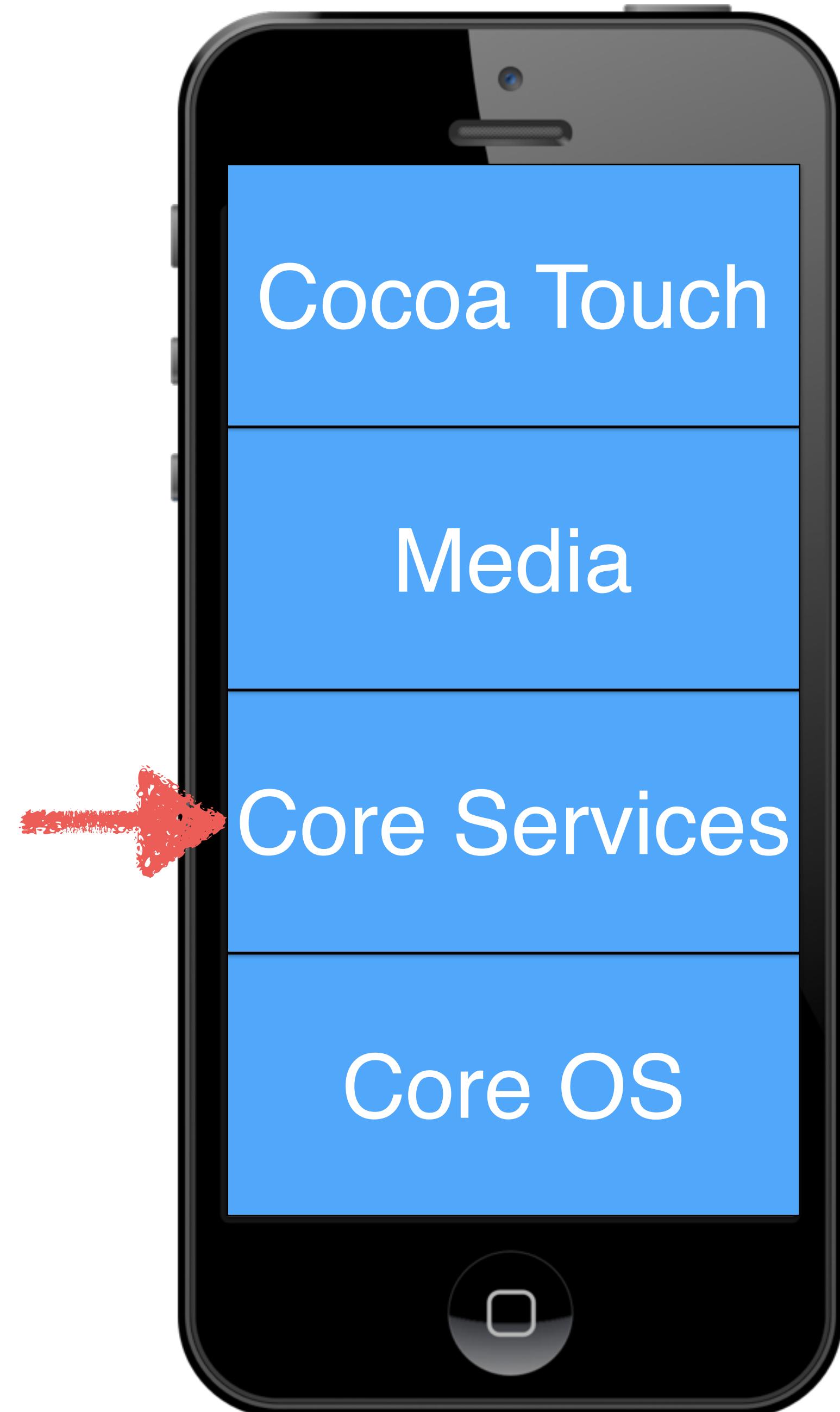
Files System

Security

Bonjous

# What is iOS

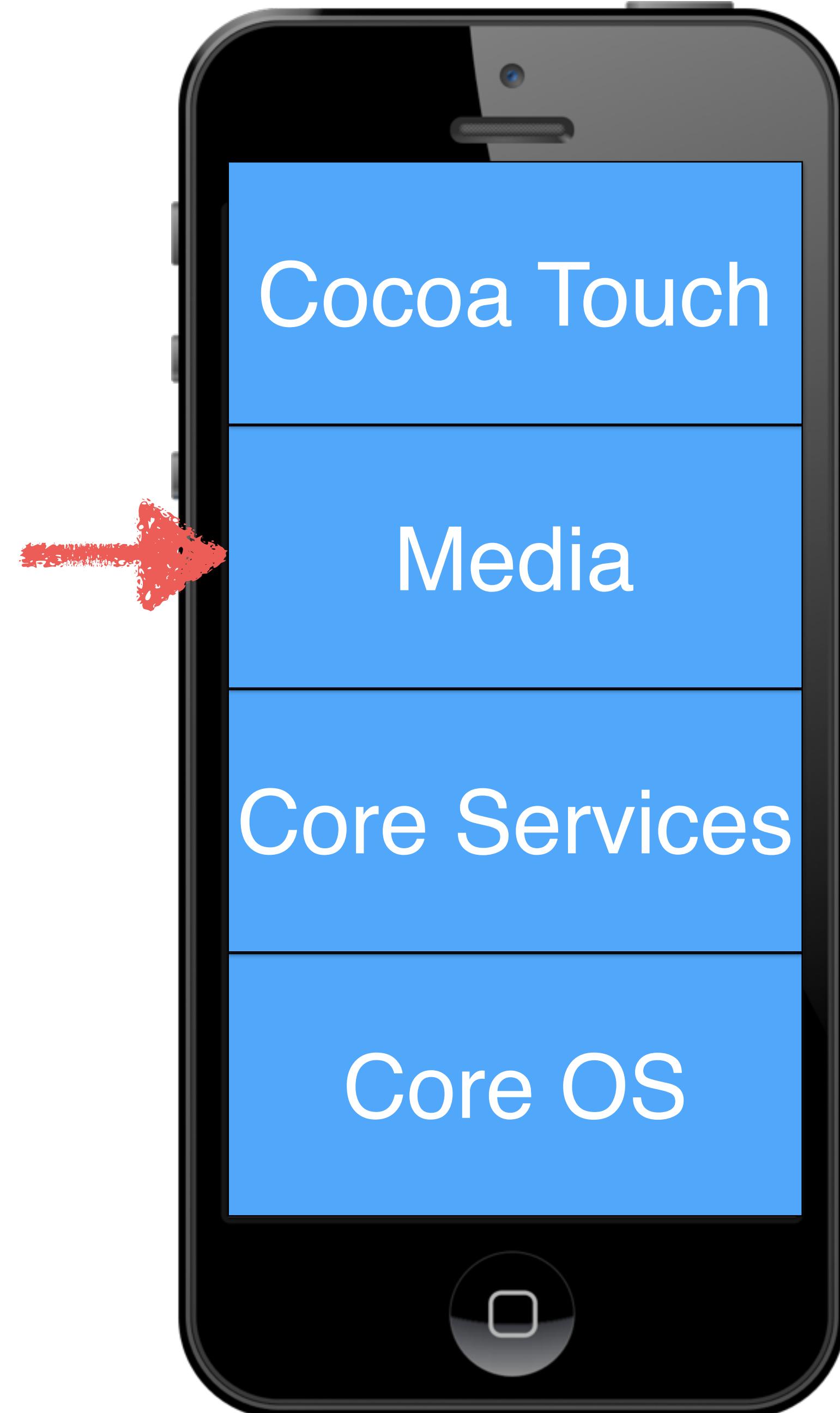
## 2. Core Services Layer



Collection	Core Location
Address Book	Net Services
Networking	Threading
File Access	Preferences
SQLite	URL Utilities

# What is iOS

## 3. Media Layer



Core Audio

Images (Png, Jpg...)

OpenAL

PDF

Audio Mixing

Quartz 2D

Audio Recording

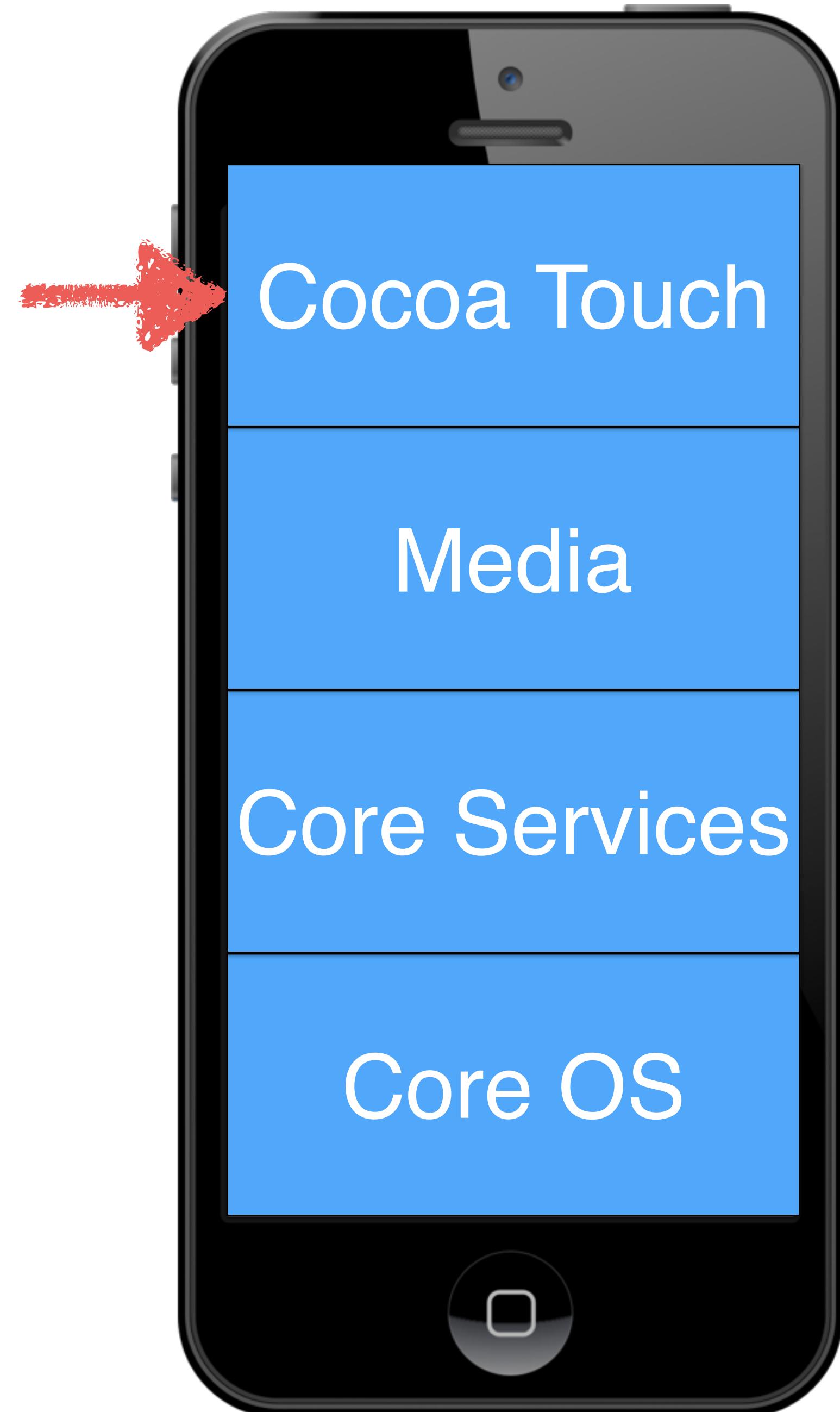
Core Animation

Videos

OpenGL ES

# What is iOS

## 4. Cocoa Touch Layer



- |                  |             |
|------------------|-------------|
| Multi-Touch      | Alerts      |
| Core Motion      | Web View    |
| View Hierarchies | MapKit      |
| Localization     | ImagePicker |
| Controls         | Camera      |



# What is a Framework

Additional code written by others

## Using Maps

- Don't write low-level code to locate our position
- Using the Mapkit framework to get current locations

# Some iOS Frameworks (1/2)

Framework	Prefix	Description
Foundation	NS	Contains interfaces for managing strings, collections, and other low-level data types.
UIKit	UI	Contains classes and methods for the iOS application user interface layer.
CoreGraphics	CG	Contains the interfaces for Quartz 2D
QuartzCore	CA	Contains the <b>Core Animation</b> interfaces.
CoreData	NS	Contains interfaces for managing your application's data model.
AVFoundation	AV	Contains Objective-C interfaces for playing and recording audio and video.
AVKit	AV	Contains Objective-C interfaces for playing and recording audio and video

Source: [Apple - iOS Frameworks](#)

# Some iOS Frameworks (2/2)

Framework	Prefix	Description
CoreBluetooth	CB	Provides access to low-power Bluetooth hardware.
CloudKit	CK	Contains Objective-C interfaces for fetching and saving iCloud data.
CoreLocation	CL	Contains the interfaces for determining a user's location.
MapKit	MK	Contains classes for embedding a map interface into your application and for reverse-geocoding coordinates.
GameKit	GK	Contains interfaces for managing peer-to-peer connectivity.
SpriteKit	SK	Facilitates the creation of sprite-based animations and rendering.
many more...	...	...

# Cocoa Touch Framework

## UIKit

- User interface elements  
(buttons, labels, tables)
- Application runtime
- Event handling
- Hardware APIs
- ...

## Foundation

- Utility classes
- Collection classes (Array, Dictionary)
- Object wrappers for system services
- ...

# MVC - Model-View-Controller 1/13

One of many possible ways to write apps

Pattern provides a way to structure and separate our code in different modules

Design  
Pattern

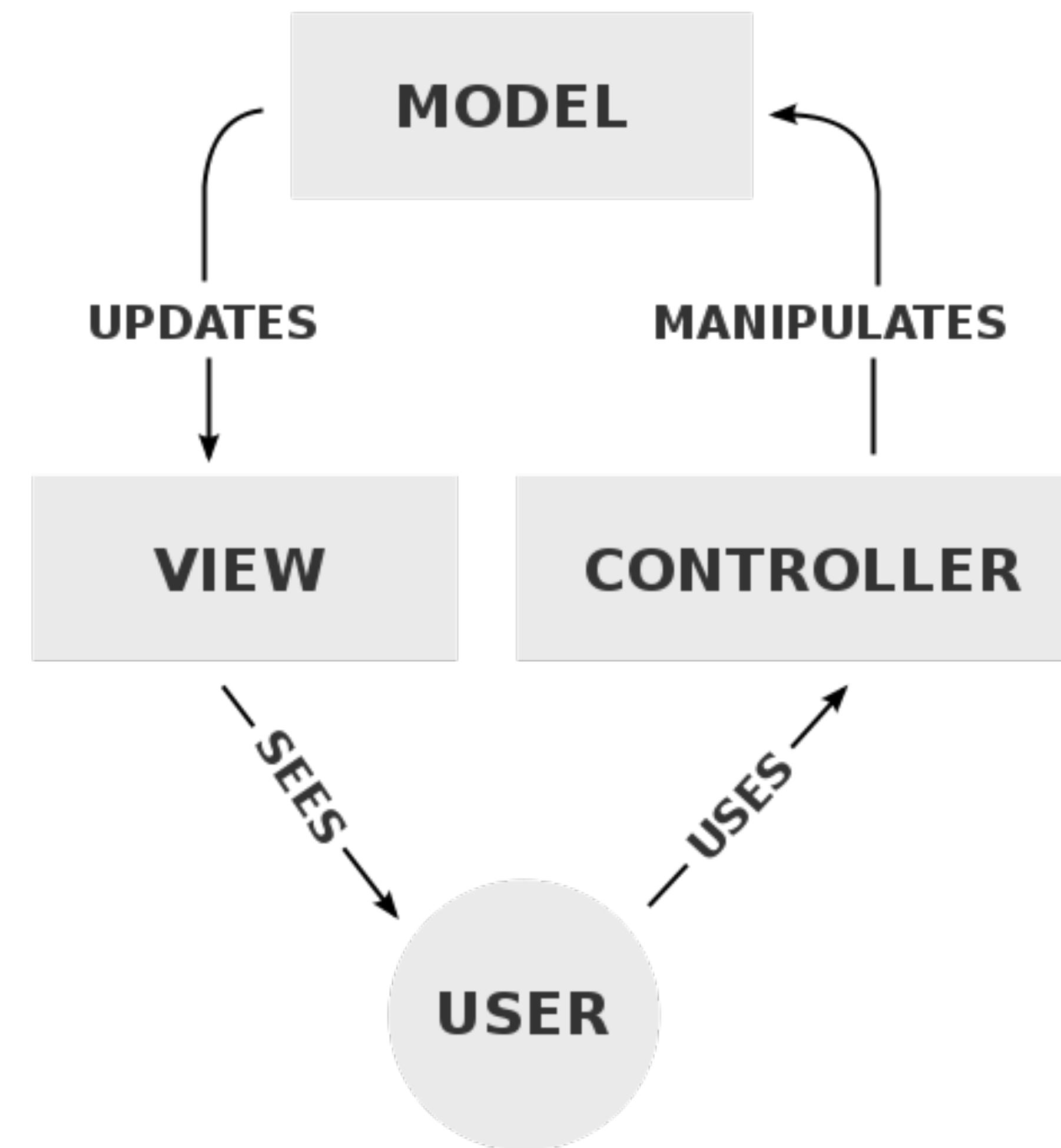
# MVC - Model-View-Controller 2/13

Well-written code is:

- Expensive
- Readable
- Maintainable over Time

Design  
Pattern

# MVC - Model-View-Controller 3/13



# MVC - Model-View-Controller 4/13



Example:  
Cat and  
Cucumbers

# MVC - Model-View-Controller 5/13

Cat

Attribute	Functions
Name	turn
Direction	accelerate
Type	deccelerate
Location	fireLaser

Cucumber

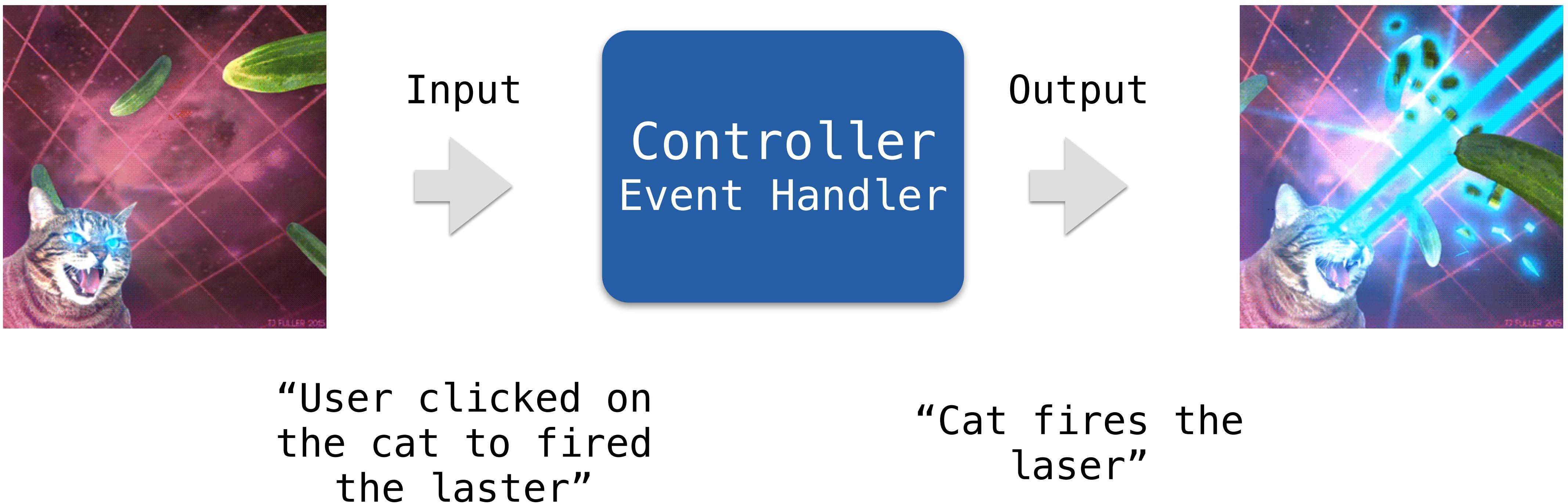
Attribute	Functions
Color	rotate
Size	explode
Radius	implode
Location	bonus

# MVC - Model-View-Controller 6/13

What an Users sees



# MVC - Model-View-Controller 7/13



# MVC - Model-View-Controller 8/13



# MVC - Model-View-Controller 9/13

Cat

Attribute	Functions
Name	Turn
Direction	Accelerate
Type	Deccelerate
Location	Fire Laser

Model  
Often represent  
as Class

# MVC - Model-View-Controller 10/13

Cat

Attribute	Functions
Name	turn
Direction	accelerate
Type	deccelerate
Location	fireLaser

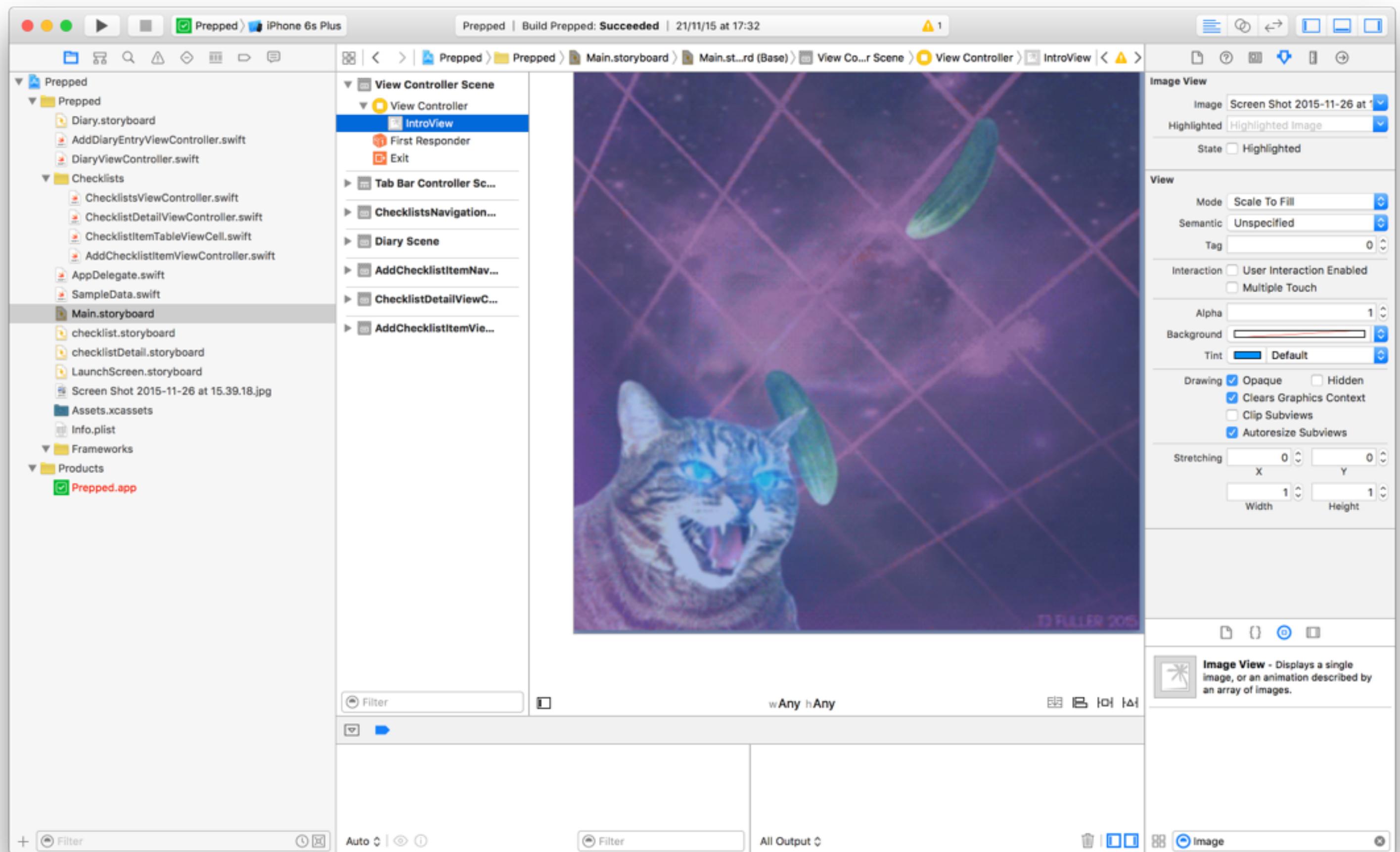
Cat.swift

```
import fundation

class Cat{
    var name:String?
    var direction:String?
    var type:String?
    var Location:location?

    func turn(...){...}
    func accelerate(...){...}
    func deccelerate(...){...}
    func fireLaser(...){...}
```

# MVC - Model-View-Controller 11/13



View  
User Interface

# MVC - Model-View-Controller 12/13

**Controller**  
A class connecting  
the model and the  
view

```
import UIKit

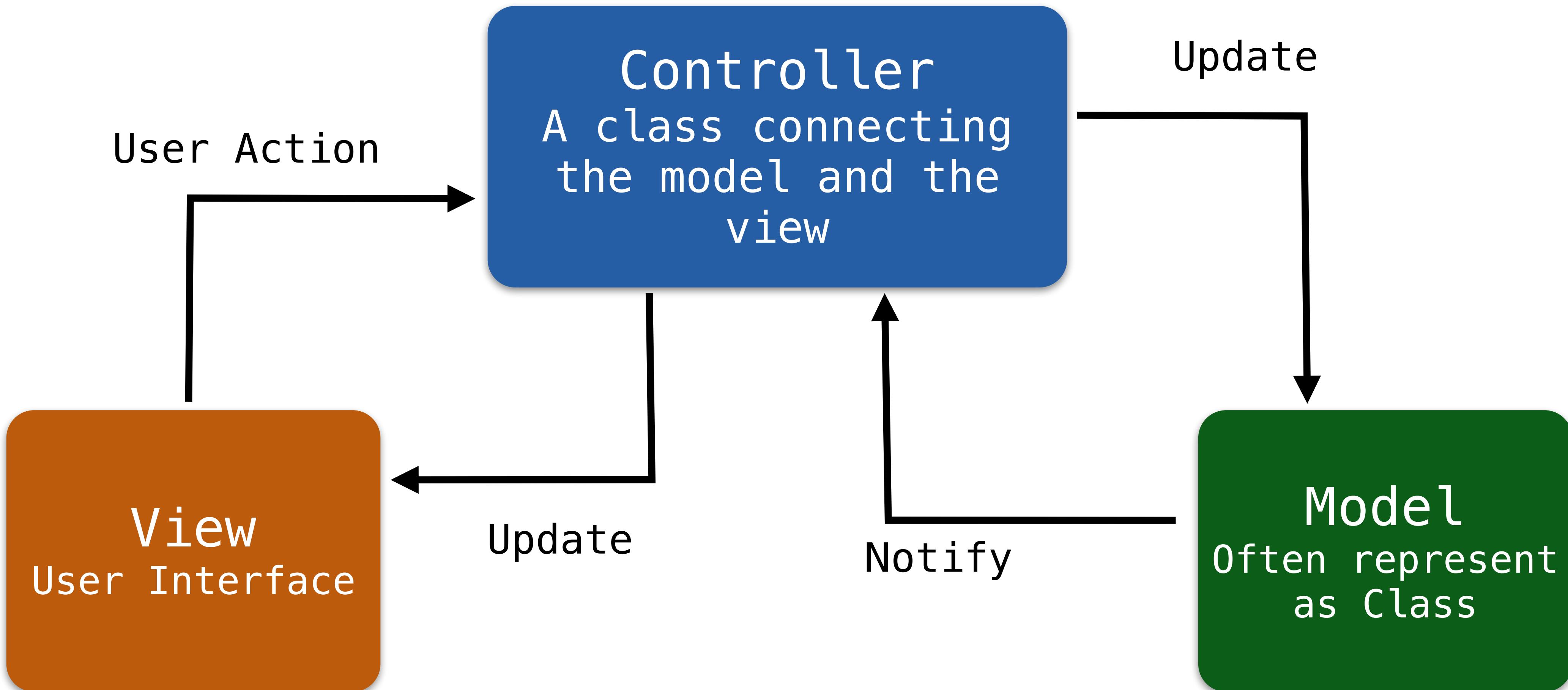
class ViewController: UIViewController {

    let cat = Cat()
    var cucumbers = [Cucumbers]()

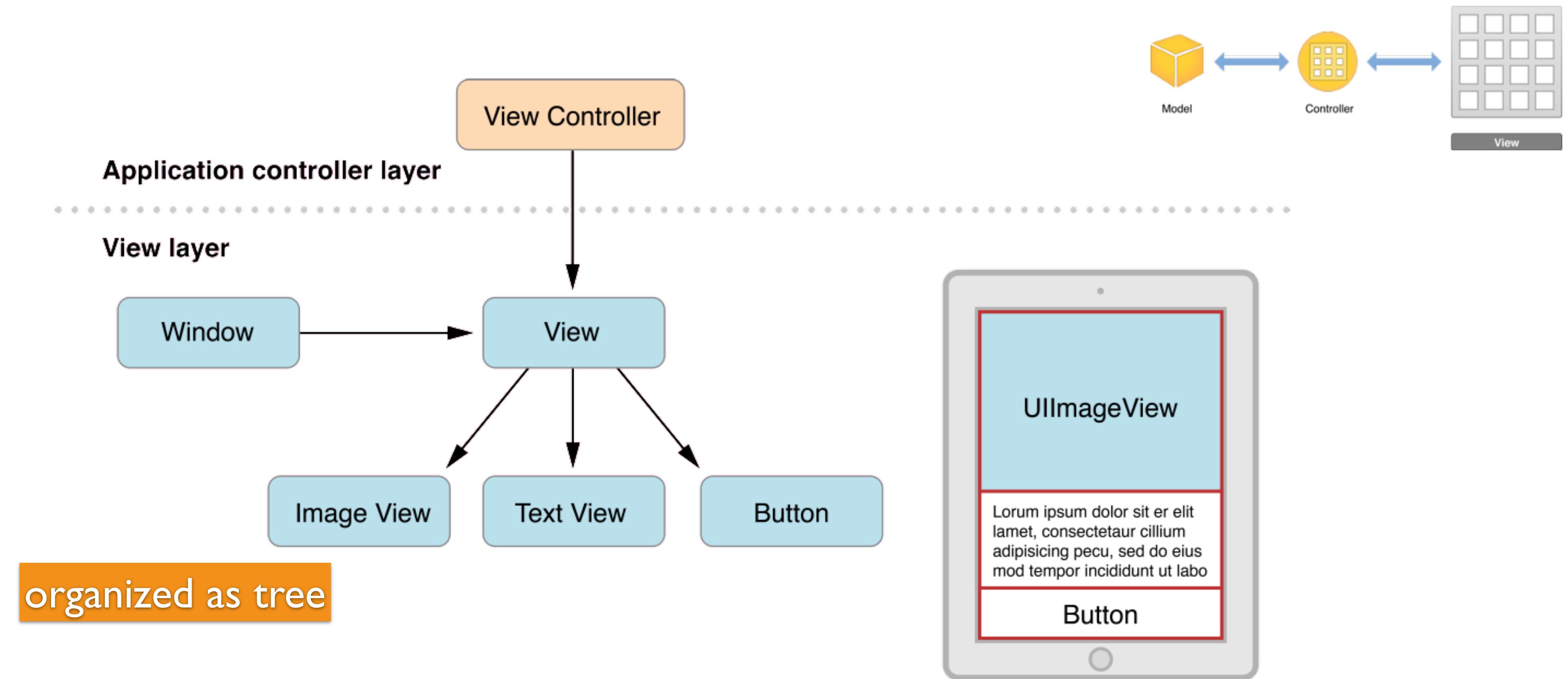
    override func viewDidLoad() {
        super.viewDidLoad()
        cat.prepareForAction()
        generateCucumbers(cucumbers)
    }

    @IBAction func swipe(sender:
UISwipeG...
```

# MVC - Model-View-Controller 13/13



# The View Hierarchy



# Windows, View Controllers, Views

- **UIWindow**

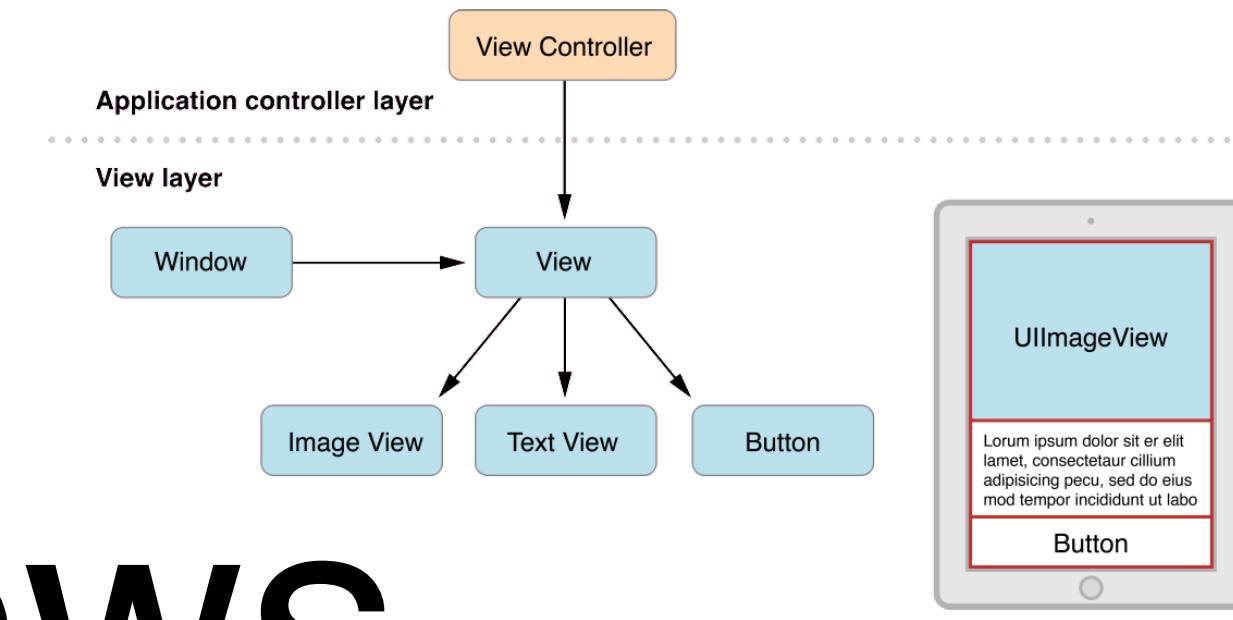
- Every App has at least one window
- like a blank window to host view controllers & views

- **UIViewController**

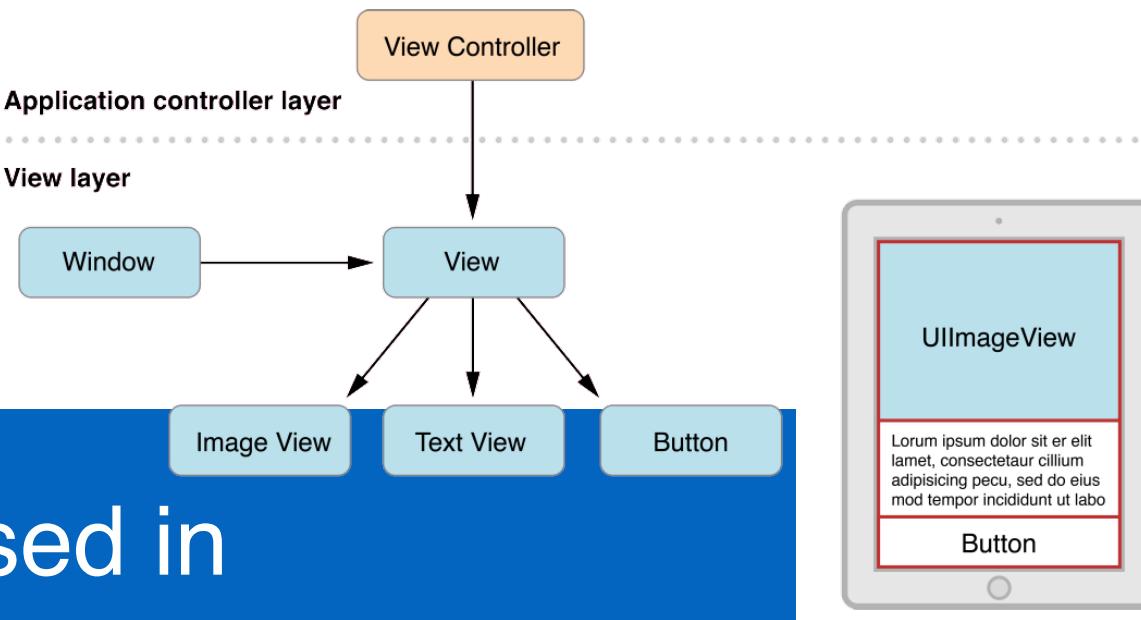
- Manages a self-contained view hierarchy

- **UIViews**

- Draws itself
- animation
- Receives events (and forwards them)



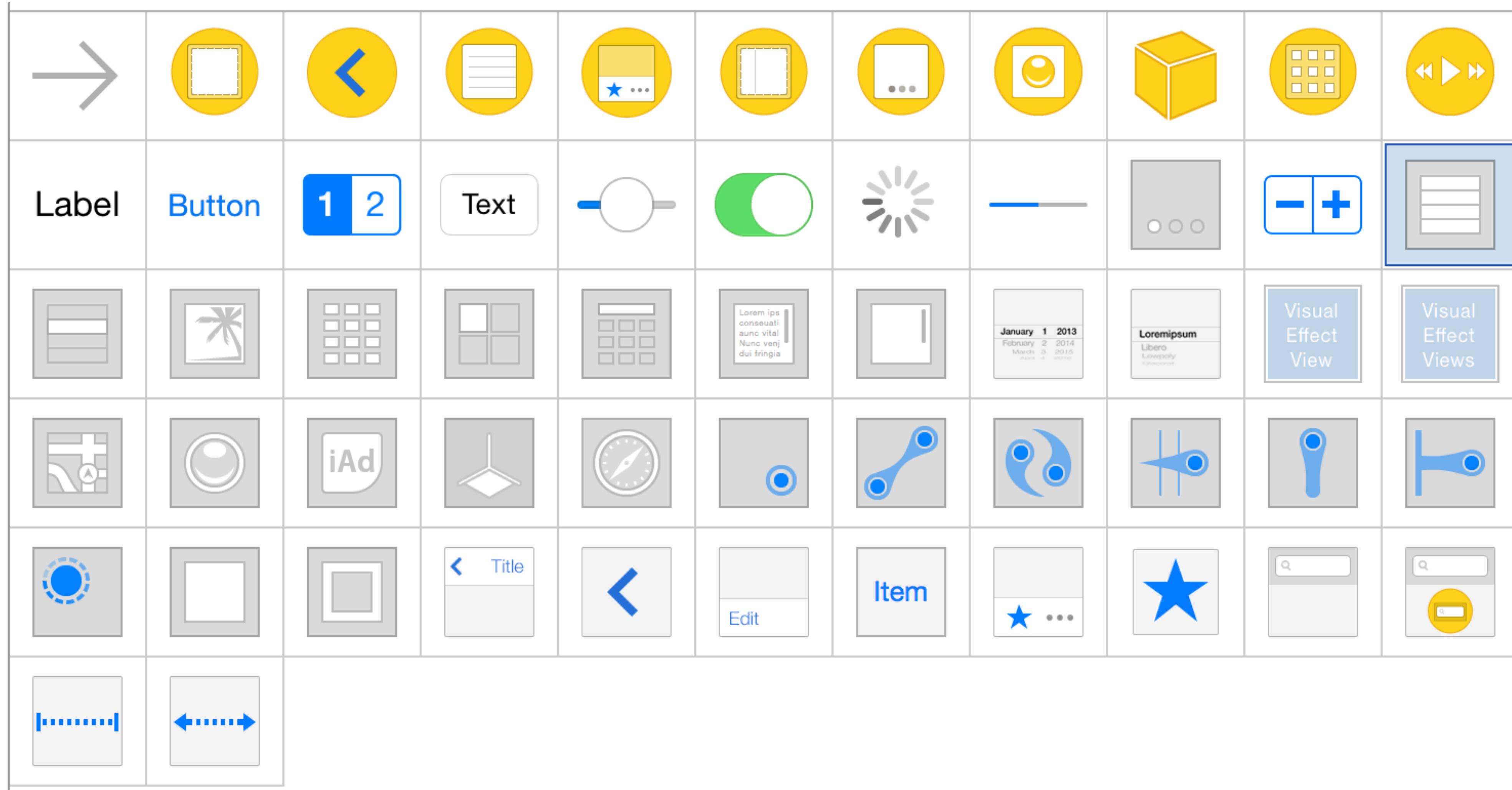
# The View Hierarchy



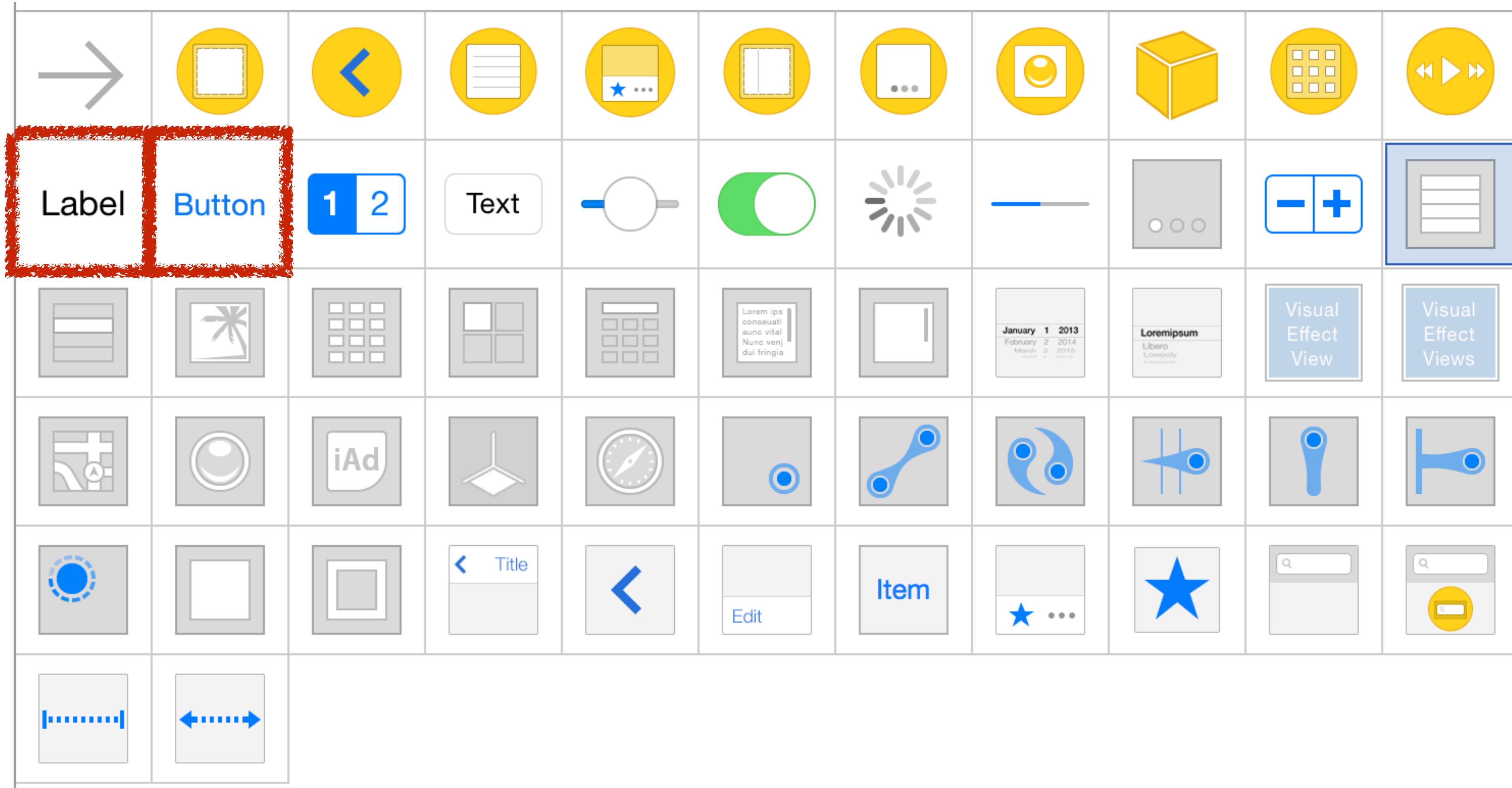
View	Purpose	Used in
Content	Display a particular type of content, such as an image or text.	<b>UIImageView</b> , <b>UILabel</b>
Collections	Display collections or groups of views.	<b>UITableView</b> , <b>UICollectionView</b>
Controls	Perform actions or display information.	<b>UIButton</b> , <b>UISlider</b> , <b>UISwitch</b>
Bars	Navigate, or perform actions.	<b>UIToolbar</b> , <b>UITabbar</b> , <b>UINavigationBar</b>
Input	Receive user input text.	<b>UISearchBar</b> , <b>UITextField</b>
Containers	Serve as containers for other views.	<b>UIView</b> , <b>UIScrollView</b>
Modal	Interrupt the regular flow of the app to allow a user to perform an action.	<b>UIActionSheet</b> , <b>UIAlertView</b>

Source: [Apple - Start Developing iOS Apps Today](#)

# The View Hierarchy

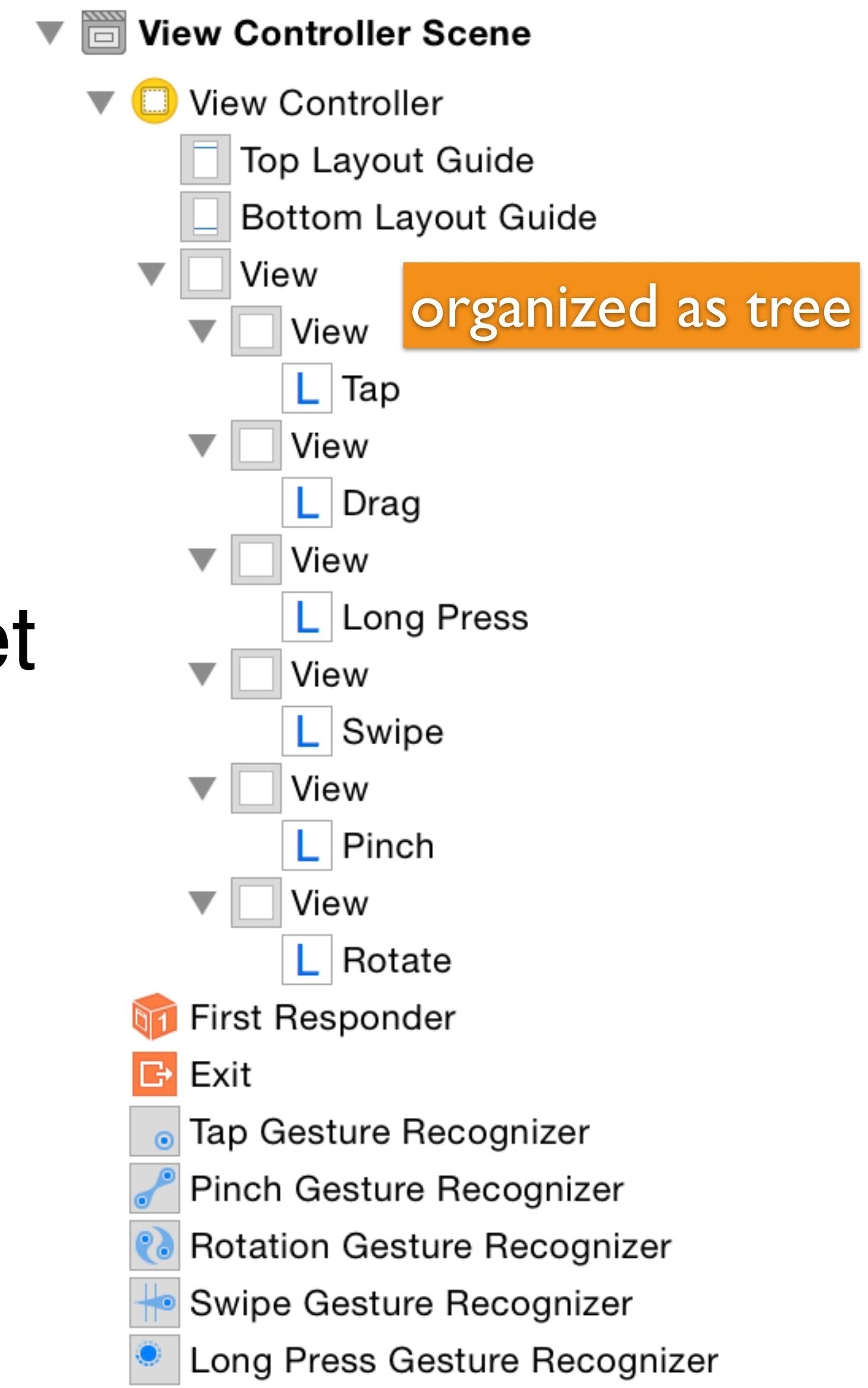


# The View Hierarchy



# The View Hierarchy

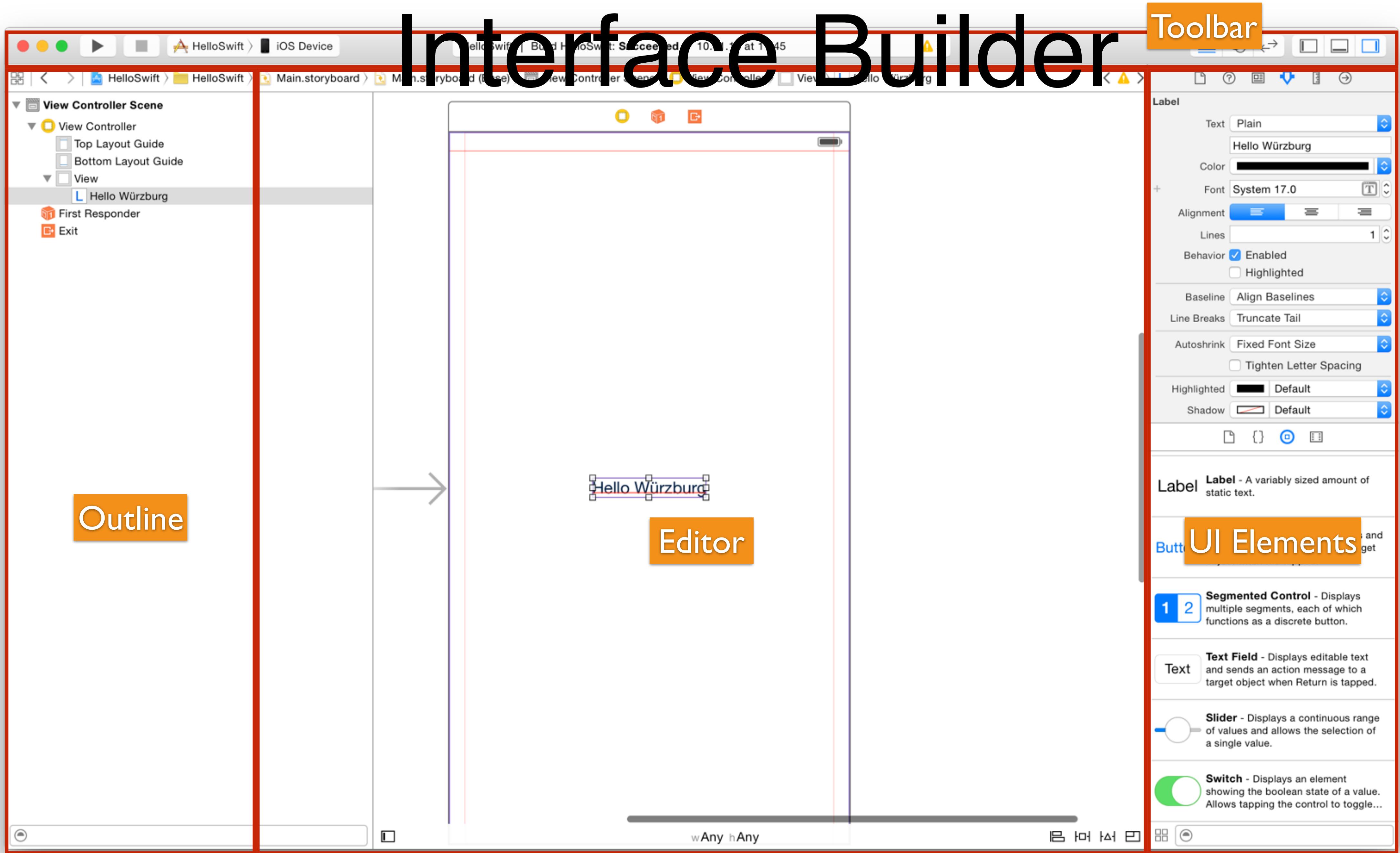
- SDK provides many types of Views to show content
- The Views are organized as tree
- Interface Builder is used to design UI in Storyboards or XIBs
- and connect the UI to code (IBAction, IBOutlet & IBOutletCollection)  
*but can be build in Code, too*
- View layout is determined by Auto Layout constraints



# Interface Builder

- is a Graphical tool to layout your User Interface
- is optional (you can layout all your Views in Code only if you prefer)
- is used to place UI elements
- sets attributes of elements
- is used to set connections between elements
- is used to wire elements to code
- stores its information in NIB files (XIB files is a valid name, too)
- uses Auto Layout as preferred layout manager

# Interface Builder



# Interface Builder

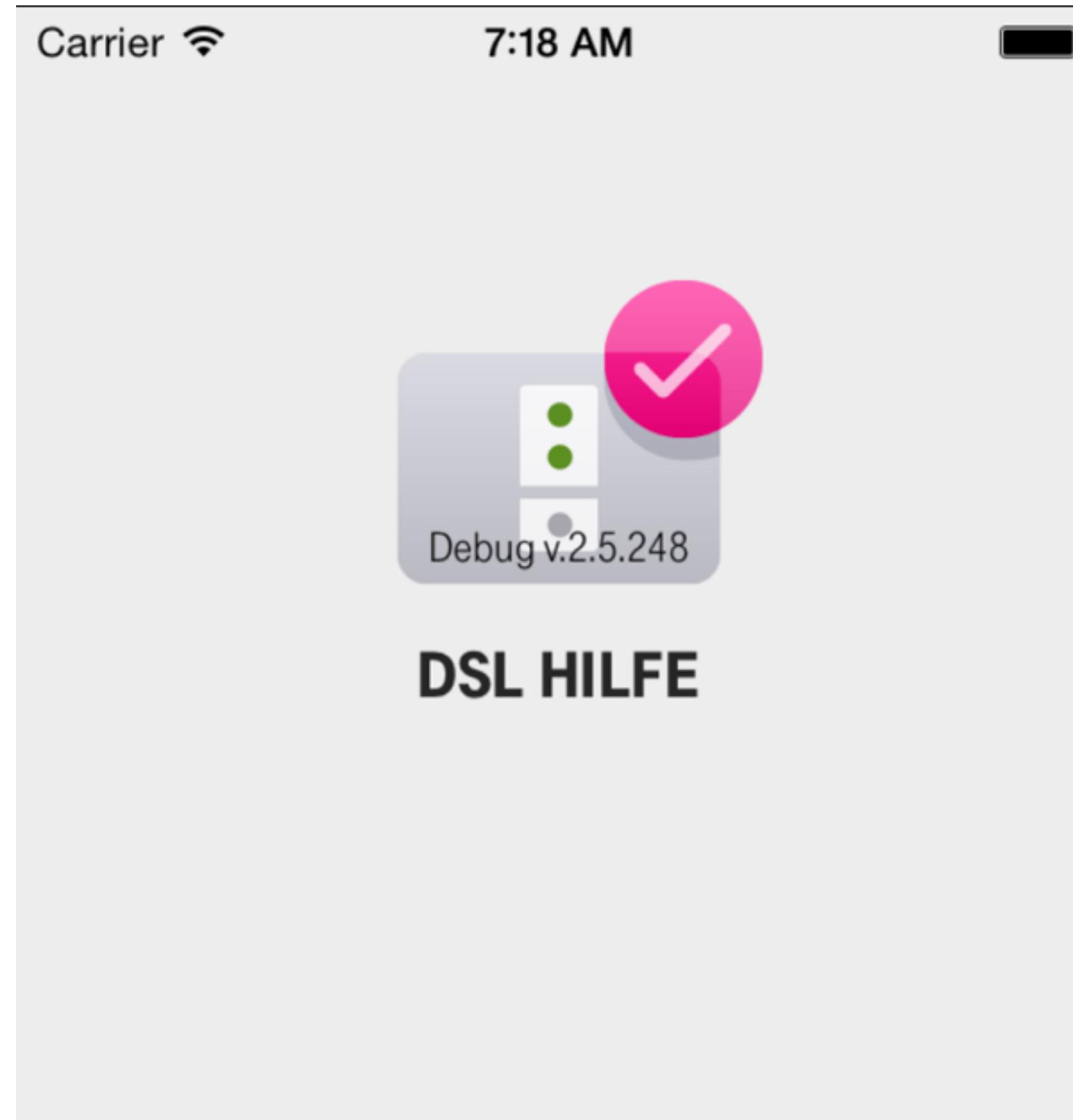


The screenshot shows the Xcode interface with the storyboard XML code. The window title is "HelloSwift" and the status bar indicates "Build HelloSwift: Succeeded" at "01.1 at 11:45". The code itself is a standard XML representation of the storyboard's structure, including scenes, objects like labels and views, and their properties.

```
initialViewController="vXZ-lx-hvc">
<dependencies>
    <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version="6711"/>
</dependencies>
<scenes>
    <!--View Controller-->
    <scene sceneID="ufC-wZ-h7g">
        <objects>
            <viewController id="vXZ-lx-hvc" customClass="ViewController" customModule="HelloSwift" customModuleProvider="target"
                sceneMemberID="viewController">
                <layoutGuides>
                    <viewControllerLayoutGuide type="top" id="jyV-Pf-zRb"/>
                    <viewControllerLayoutGuide type="bottom" id="2fi-mo-0CV"/>
                </layoutGuides>
                <view key="view" contentMode="scaleToFill" id="kh9-bI-dsS">
                    <rect key="frame" x="0.0" y="0.0" width="414" height="736"/>
                    <autoresizingMask key="autoresizingMask" flexibleMaxX="YES" flexibleMaxY="YES"/>
                    <subviews>
                        <label opaque="NO" userInteractionEnabled="NO" contentMode="left" horizontalHuggingPriority="251"
                            verticalHuggingPriority="251" fixedFrame="YES" text="Hello Würzburg"
                            lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
                            adjustsFontSizeToFit="NO" translatesAutoresizingMaskIntoConstraints="NO"
                            id="9uE-ro-kQH">
                            <rect key="frame" x="148" y="357" width="118" height="21"/>
                            <fontDescription key="fontDescription" type="system" pointSize="17"/>
                            <color key="textColor" red="0.0" green="0.0" blue="0.0" alpha="1" colorSpace="calibratedRGB"/>
                            <nil key="highlightedColor"/>
                        </label>
                    </subviews>
                    <color key="backgroundColor" white="1" alpha="1" colorSpace="custom" customColorSpace="calibratedWhite"/>
                </view>
                <simulatedScreenMetrics key="simulatedDestinationMetrics" type="retina55"/>
            </viewController>
            <placeholder placeholderIdentifier="IBFirstResponder" id="x5A-6p-PRh" sceneMemberID="firstResponder"/>
        </objects>
    </scene>
</scenes>
</document>
```

Storyboards/XIBs are basically plain XML-files

# Tree Example - DSL Hilfe



Anschluss einrichten



Fehler beheben



Speedport einstellen



DSL Hilfe (iPhone Simulator)

UIScreen



# Tree Example - DSL Hilfe



DSL Hilfe 248  
iPhone Simulator (iOS 8.1)

UIWindow

- UILayoutContainerView
- UINavigationTransitionView
- UIViewControllerWrapperView
- UIView
  - Constraints
- UITableView
  - Constraints
- UITableViewController
- UITableViewCellSeparatorView
- UITableCellContentView
- Constraints
- UILabel: "Speedport"
- UIImageView
- UITableViewCellSeparatorView
- UITableCellContentView
- Constraints
- UILabel: "Fehler beheben"
- UIImageView
- UITableViewCellSeparatorView
- UITableCellContentView
- Constraints
- UILabel: "Anschluss"
- UIImageView
- UITableViewCellSeparatorView
- UITableCellContentView
- Constraints
- UILabel: "Anleitungen & Videos"
- UIImageView
- UITableViewCellSeparatorView
- UITableCellContentView
- Constraints
- UILabel: "Info & Kontakt"
- UIImageView

DSL\_TemplateA1\_A2\_BottomTo

- Constraints
- UIView

UILabel

**Text:** Fehler beheben

**Color:** Current Color

#FF262626

**Font:** Tele-GroteskNormal

Regular

18

**Alignment:** Left

**Lines:** 1

Enabled

**Shadow:**

**Color:** None (Nil)

**Offset:** W 0 H -1

**Highlight:**

**Color:** None (Nil)

Highlighted

**Baseline:** Align Baselines

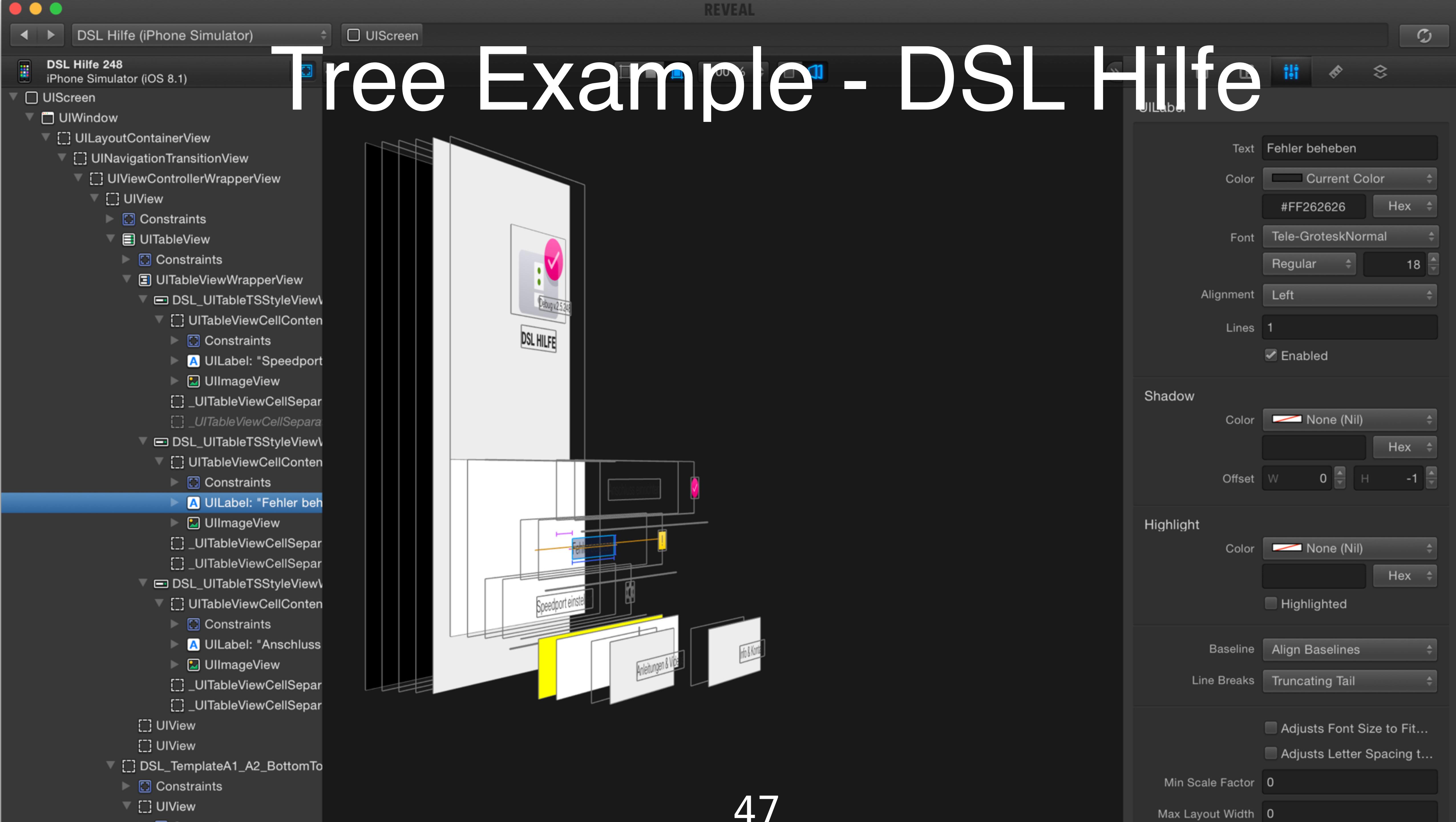
**Line Breaks:** Truncating Tail

Adjusts Font Size to Fit...

Adjusts Letter Spacing t...

**Min Scale Factor:** 0

**Max Layout Width:** 0



▼ Part IV: UI Elements

▼ Bars

The Status Bar

Navigation Bar

Toolbar

Toolbar and Navigation Bar B...

Tab Bar

Tab Bar Icons

Search Bar

Scope Bar

▼ Content Views

Activity

Activity View Controller

Collection View

Container View Controller

Image View

Map View

Page View Controller

Popover

ScrollView

Split View Controller

Table View

Text View

WebView

▼ Controls

Activity Indicator

Contact Add Button

Date Picker

Detail Disclosure Button

Info Button

Label

Network Activity Indicator

Page Control

Picker

Progress View

Refresh Control

Rounded Rectangle Button

Segmented Control

# Finding the right View

## Text View

A **text view** accepts and displays multiple lines of attributed text.

◀ [UICatalog](#) **Text View**

This is a text view that uses attributed text. You can use text attributes such as **bold**, **highlighted**, **underlined**, and **tinted**.

API NOTE

To learn more about defining a text view in your code, see [Text Views](#) in [UIKit User Interface Catalog](#).

A text view:

- Is a rectangle of any height
- Supports scrolling when the content is too large to fit inside its bounds
- Supports custom fonts, colors, and alignments (by default, a text view displays left-aligned system font in black)
- Can support editing, in which case a keyboard appears when the user taps inside the text view (the keyboard's input method and layout are determined by the text view's settings)

⌘ + Shift + 0 —> “Part IV: UI Elements”

**Always make sure the text is easy to read.** Although you can use attributed strings to combine multiple fonts, colors, and alignments in creative ways, it's important to consider the readability of the text. It's a good idea to support Dynamic Type and use the `UIFont.preferredFontForTextStyle` to get the font for display in different situations.

# @IBAction

The screenshot shows the Xcode interface with a storyboard and a Swift code editor. A red arrow points from the storyboard to the right-click context menu, which is open over a button labeled "Hello". Another red arrow points from the storyboard to the code editor, specifically to the declaration of the `swiftAction` method. A blue callout box highlights the `@IBAction func myAction:(id)sender` line in the code.

```
// Copyright (c) 2015 Flori & Andy. All rights reserved.  
//  
import UIKit  
  
class ViewController: UIViewController {  
  
    @IBOutlet weak var helloLbl: UILabel!  
  
    @IBAction func helloAction(sender: UIButton) {  
    }  
  
    @IBAction func swiftAction(sender: UIButton) {  
    }  
}
```

or right click on element

# @IBOutlet

The screenshot shows the Xcode interface with a blue callout box containing the text "@IBOutlet weak var myOutlet". A red arrow points from this box to the "Referencing Outlets" section of the Attributes Inspector for a UILabel named "helloLbl". Another red arrow points from the text "or right click on element" to the "Hello Würzburg" button in the storyboard. The top bar shows "HelloSwift" is running on an iPhone 6. The bottom status bar indicates "wAny hAny".

```
// Copyright (c) 2015 Flori & Andy. All rights reserved.  
//  
import UIKit  
  
class ViewController: UIViewController {  
  
    @IBOutlet weak var helloLbl: UILabel!  
  
    @IBAction func helloAction(sender: UIButton) {  
    }  
  
    @IBAction func swiftAction(sender: UIButton) {  
    }  
}
```

# Application Components

## AppDelegate.swift

- UIApplication
- CFRunLoop

## Info.plist

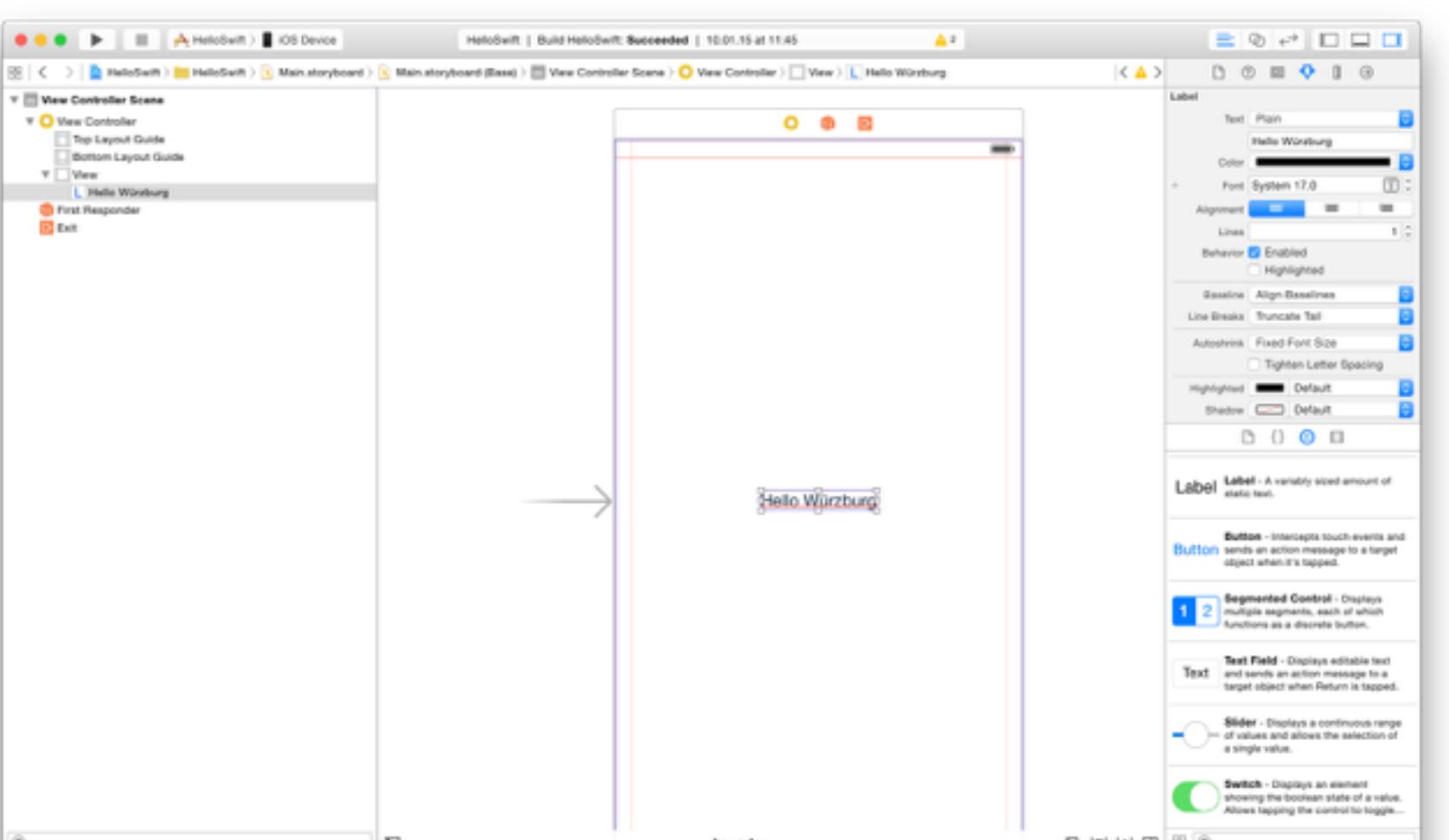
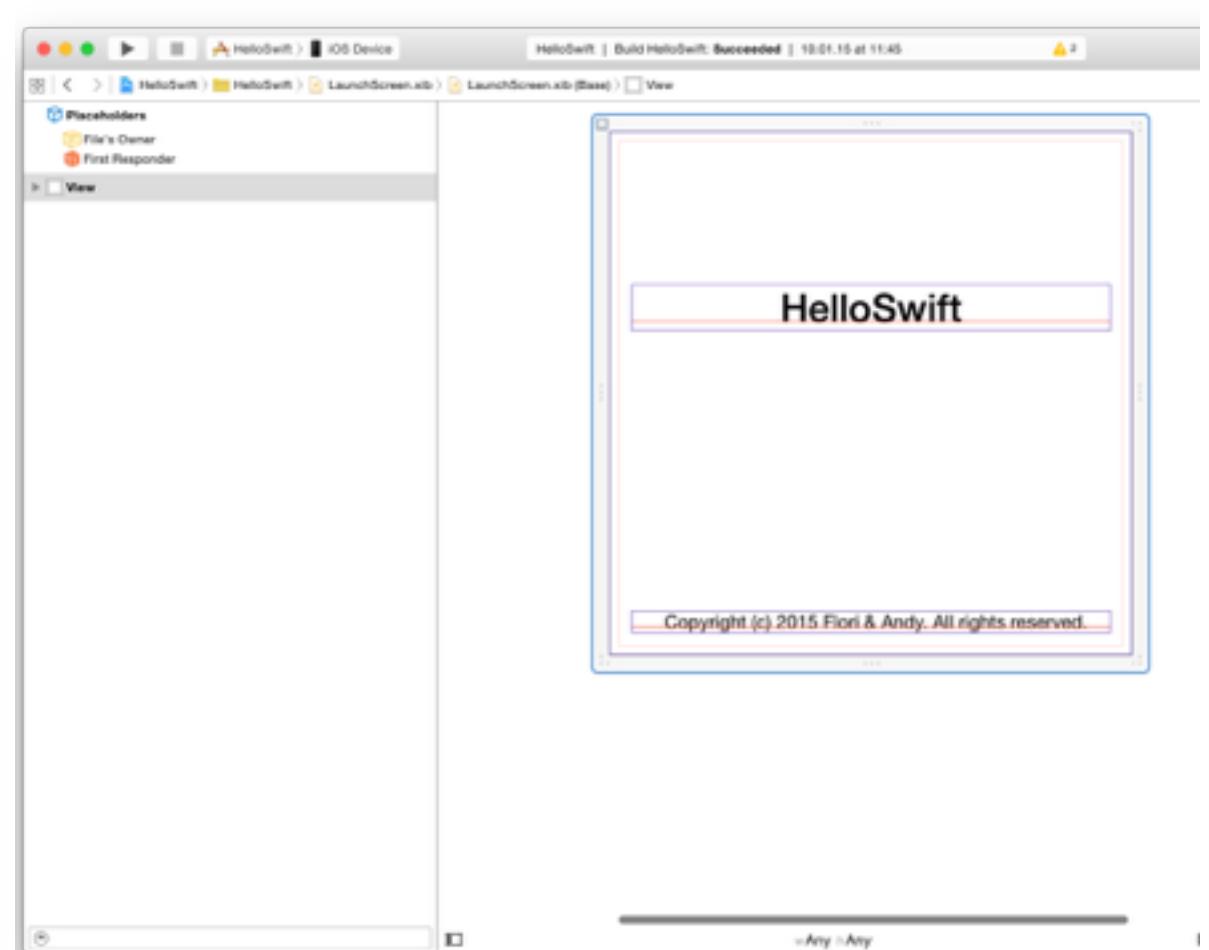
## MainStoryboard

## LaunchScreen.xib

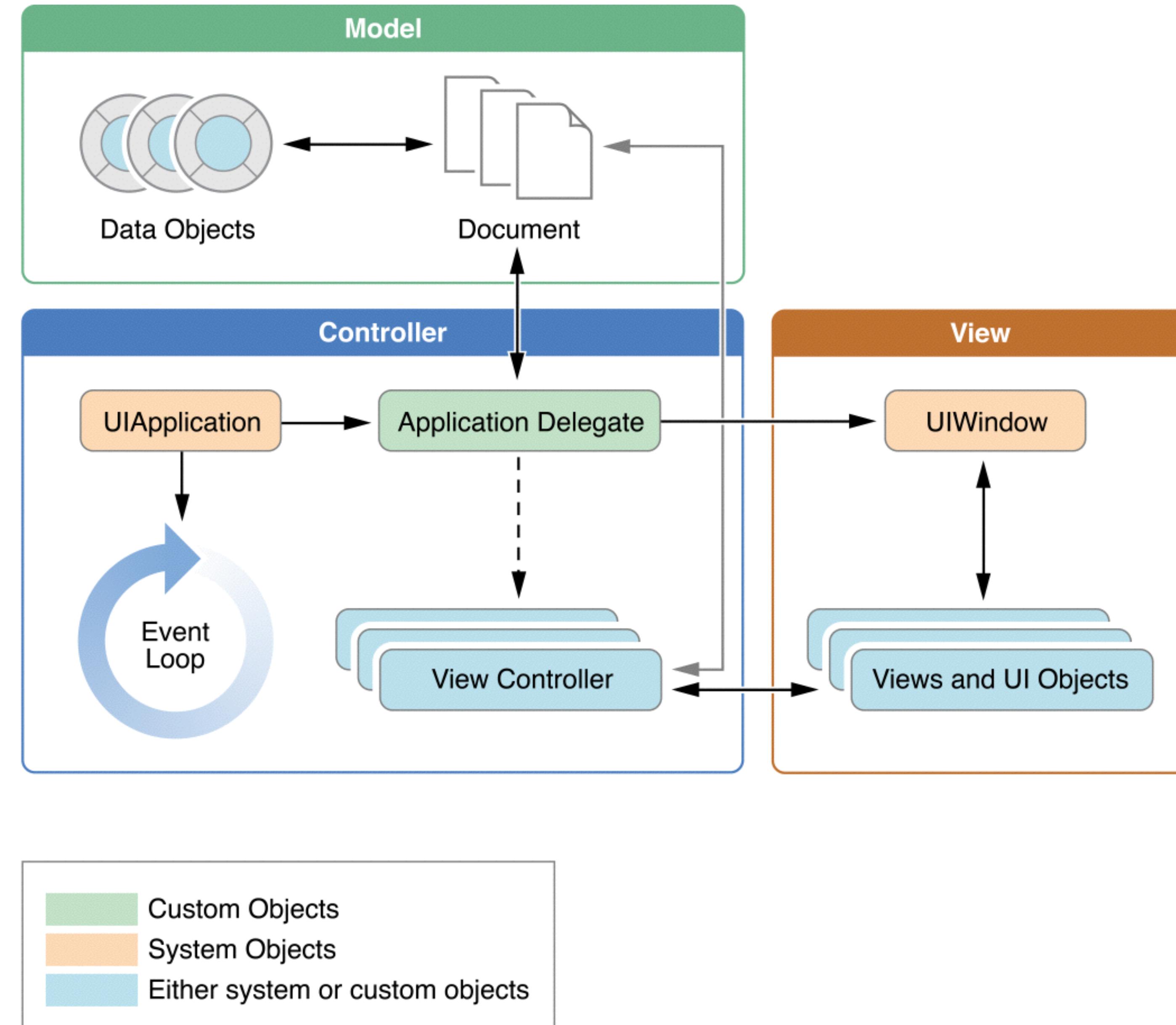
## ViewController.swift

Key	Type	Value
▼ Information Property List		
Localization native development r...	String	en
Executable file	String	\$(EXECUTABLE_NAME)
Bundle identifier	String	de-iosprogrammingwithswift.\$(PRODUCT_NAME:rfc1034identifier)
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1
Application requires iPhone envir...	Boolean	YES
Launch screen interface file base...	String	LaunchScreen
Main storyboard file base name	String	Main
► Required device capabilities	Array	(1 item)
► Supported interface orientations	Array	(3 items)

```
1 // AppDelegate.swift
2 // HelloSwift
3 // Copyright (c) 2015 Flori & Andy. All rights reserved.
4
5 import UIKit
6
7 @UIApplicationMain
8 class AppDelegate: UIResponder, UIApplicationDelegate {
9
10     var window: UIWindow?
11
12     func application(application: UIApplication, didFinishLaunchingWithOptions: [NSObject: AnyObject]?) -> Bool {
13         // Override point for customization after application launch.
14         return true
15     }
16
17     func applicationWillResignActive(application: UIApplication) {
18         // Sent when the application is about to move from active to inactive state. This can occur for
19         // certain types of temporary interruptions (such as an incoming phone call or SMS message) or when
20         // the user quits the application and it begins the transition to the background state.
21         // Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame rates.
22         // Games should use this method to pause the game.
23     }
24
25     func applicationDidEnterBackground(application: UIApplication) {
26         // Use this method to release shared resources, save user data, invalidate timers, and store enough
27         // application state information to restore your application to its current state in case it is
28         // terminated later.
29         // If your application supports background execution, this method is called instead of
30         // applicationWillTerminate: when the user quits.
31     }
32
33     func applicationWillEnterForeground(application: UIApplication) {
34         // Called as part of the transition from the background to the inactive state; here you can undo many
35         // changes made on entering the background.
36     }
37 }
```

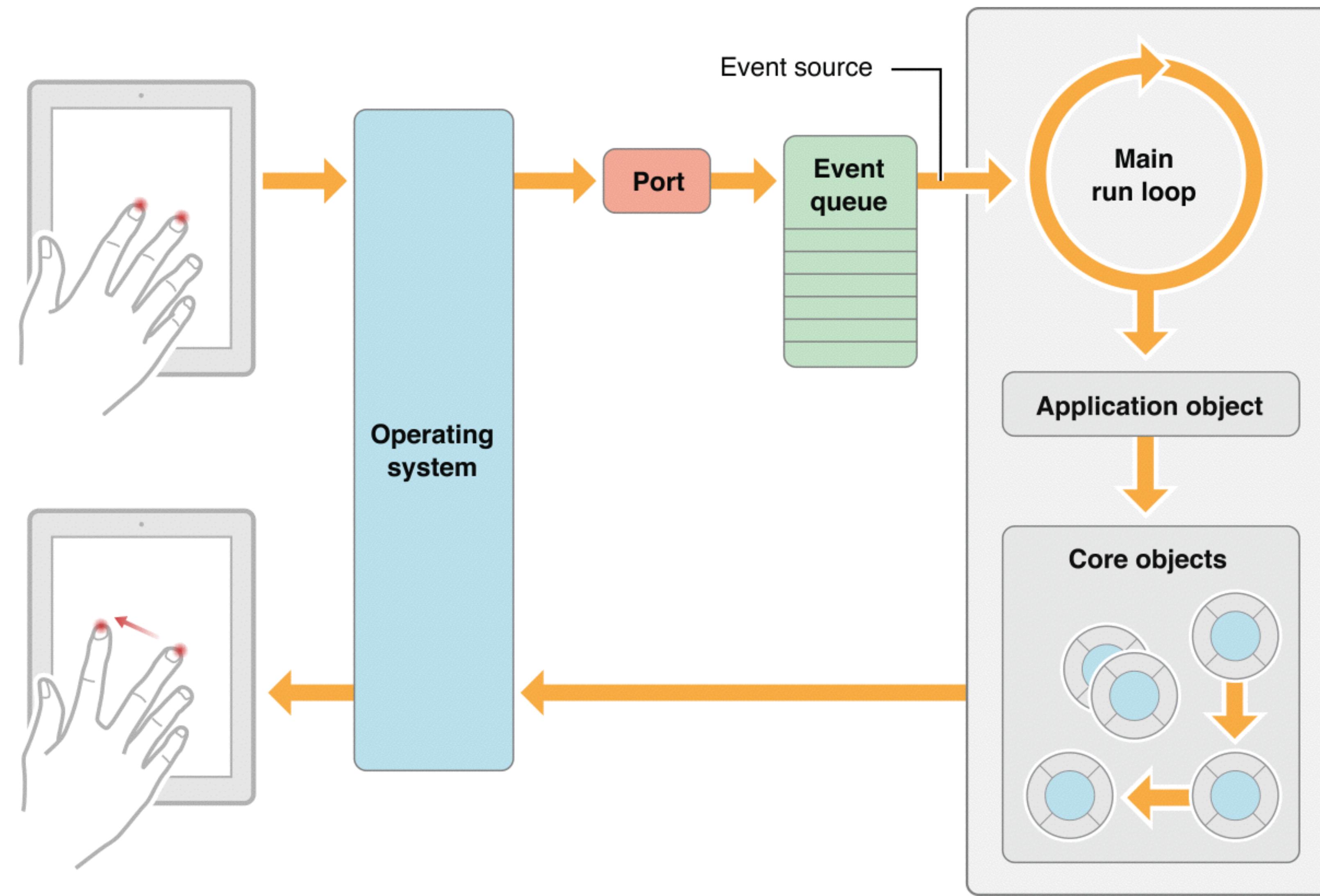


# App Life Cycle

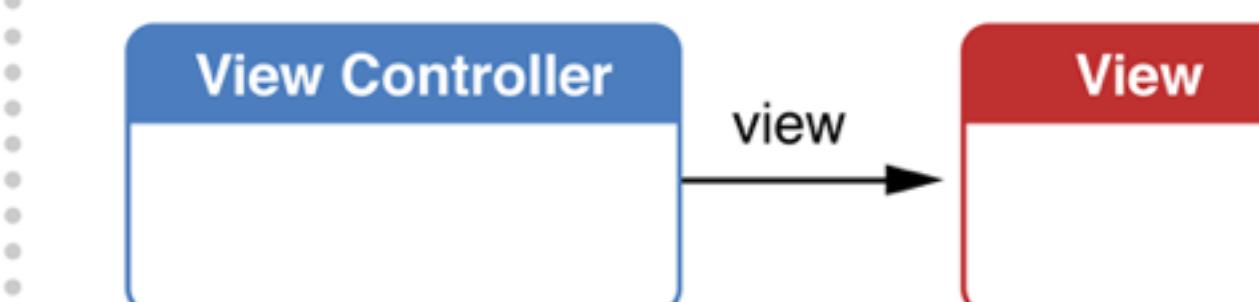
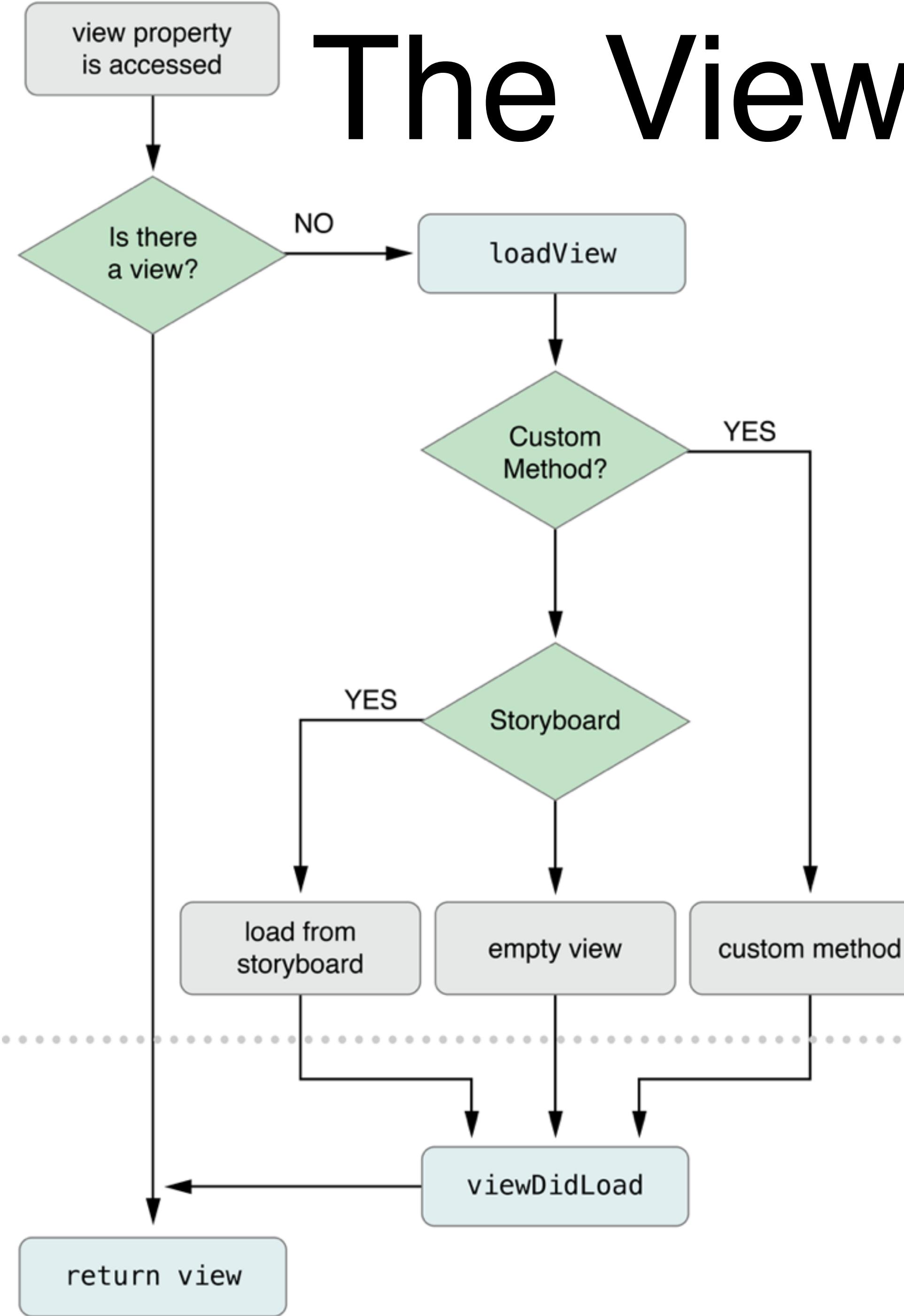


Source: [Apple - The App Life Cycle](#)

# The Main Run Loop

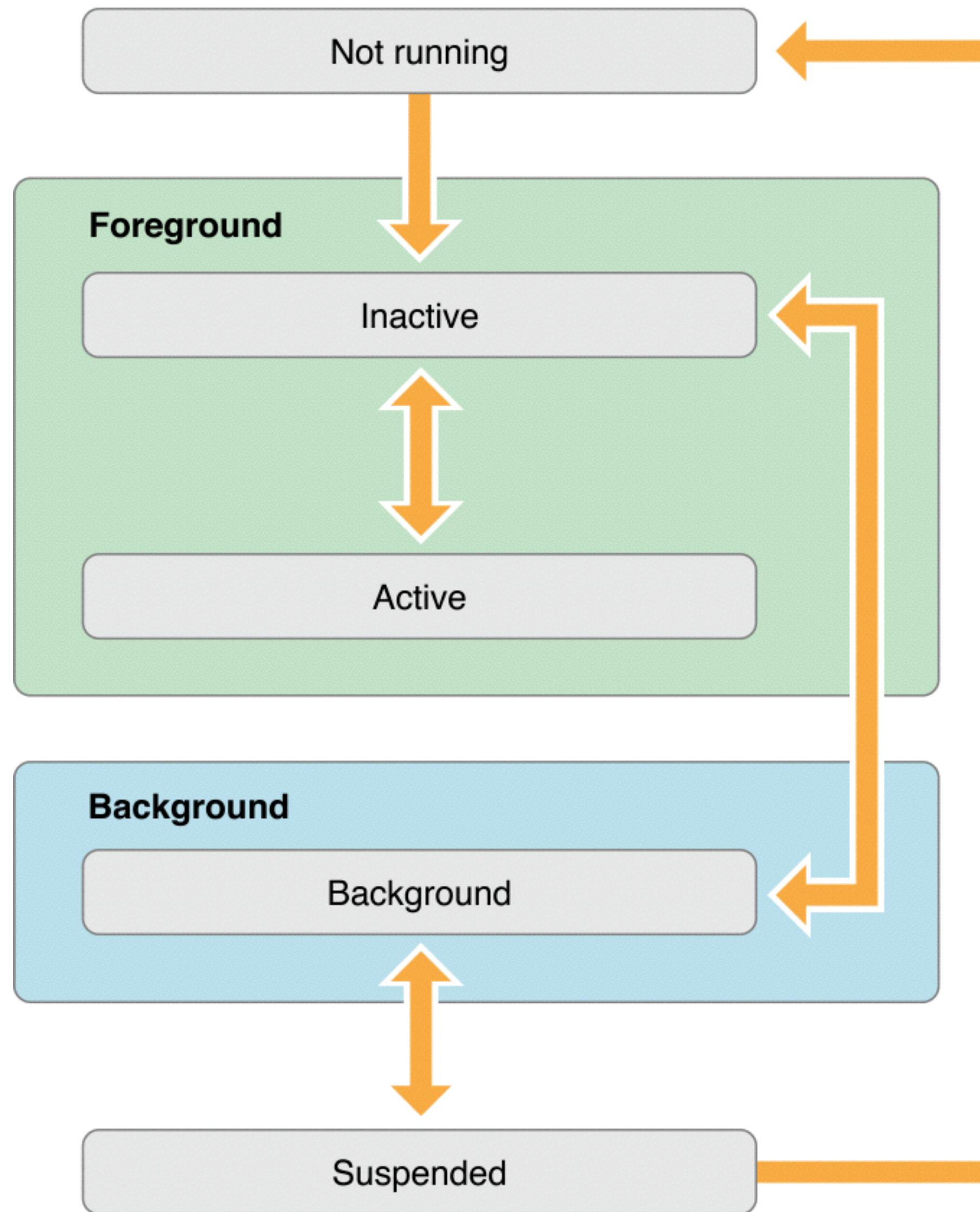


# The View Life Cycle

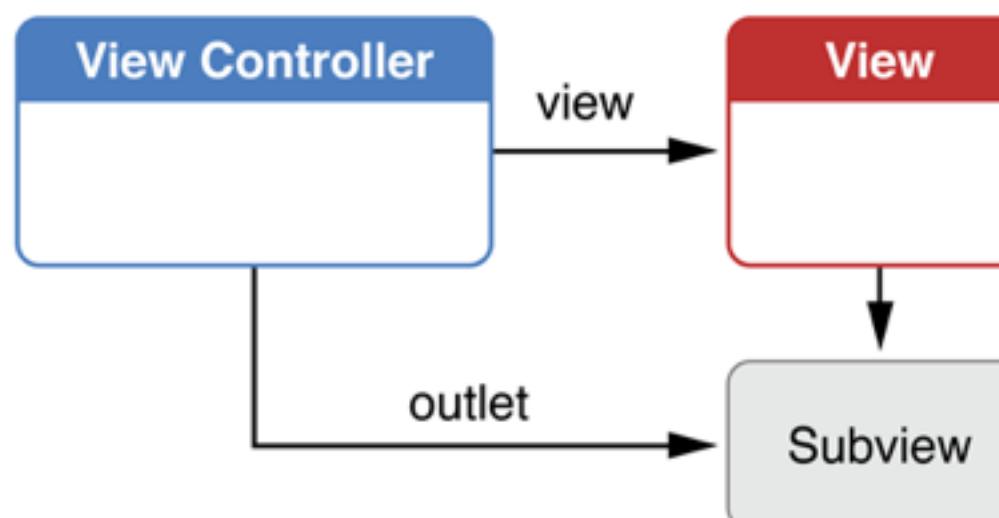
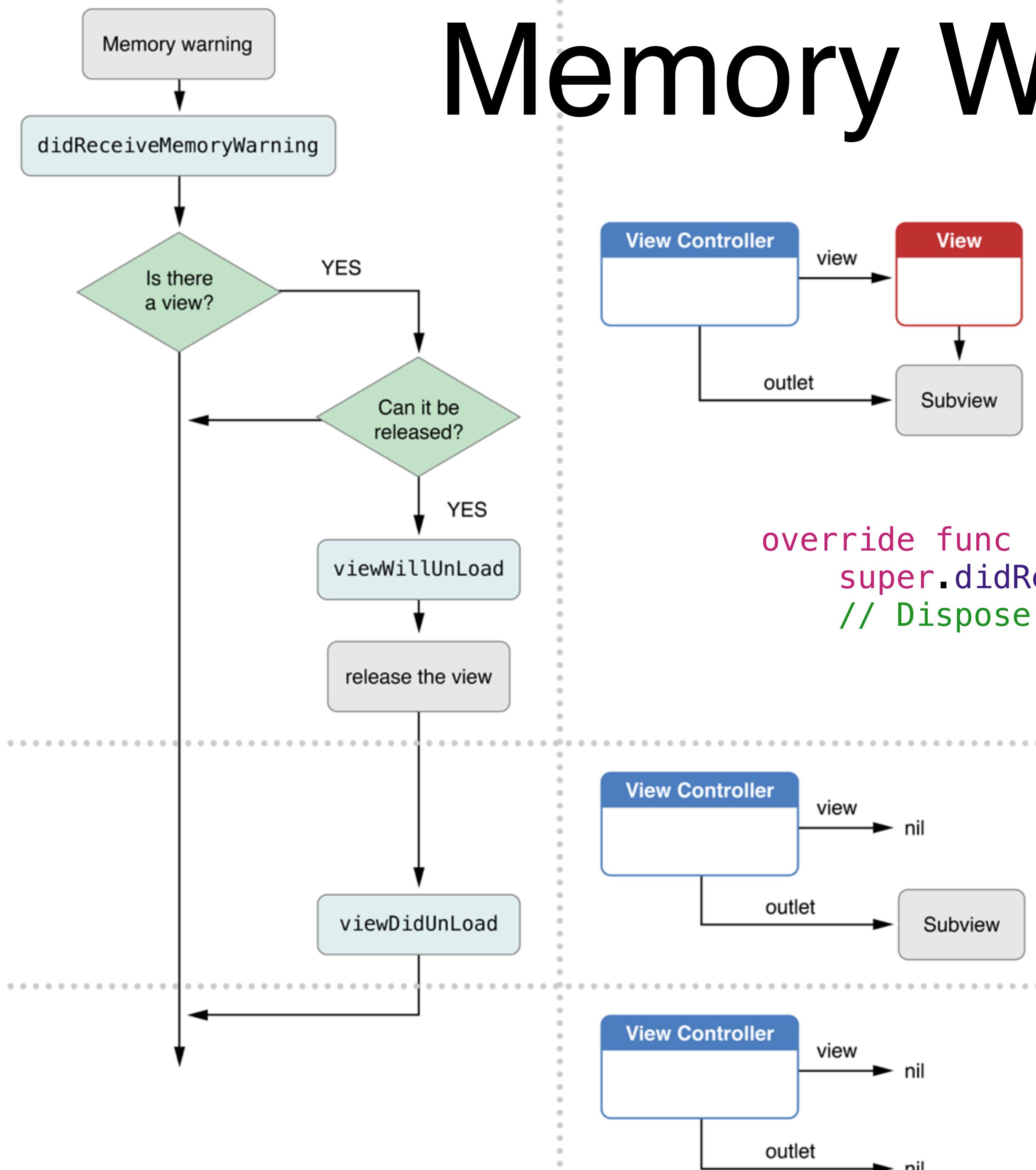


Source: [View Controller Programming Guide for iOS](#)

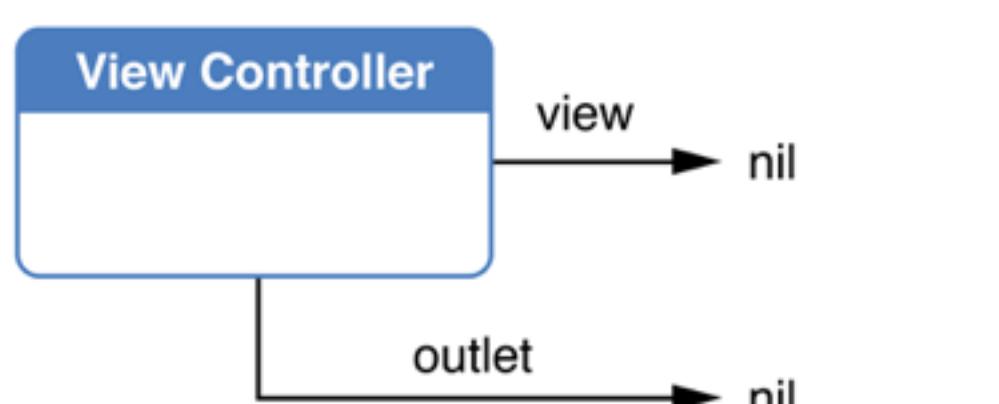
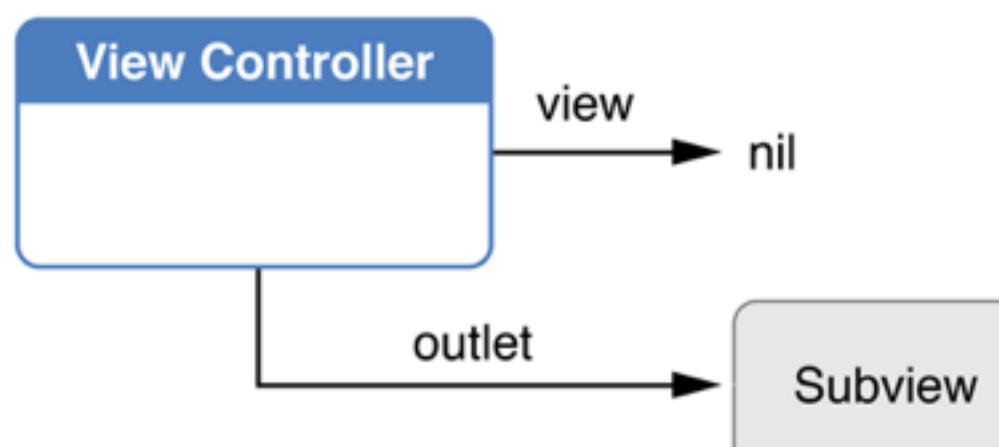
# Execution States for Apps



# Memory Warnings



`override func  
super.didReceiveMemoryWarning()  
// Dispose of any resources that can be recreated.`



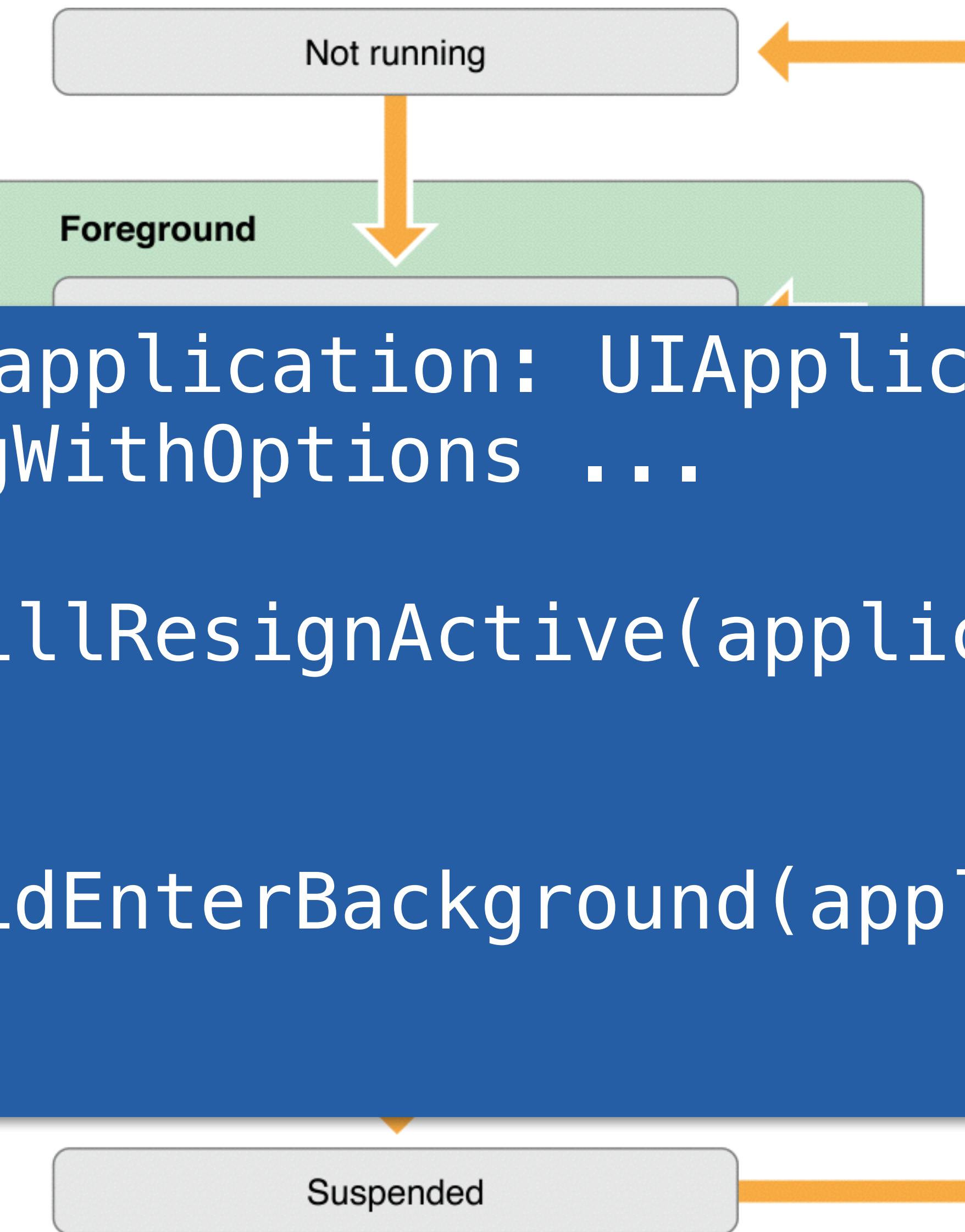
# Execution States for Apps

AppDelegate.swift

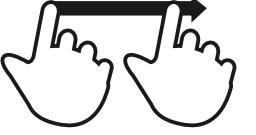
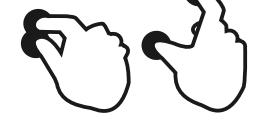
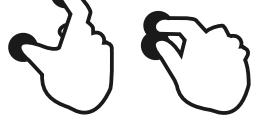
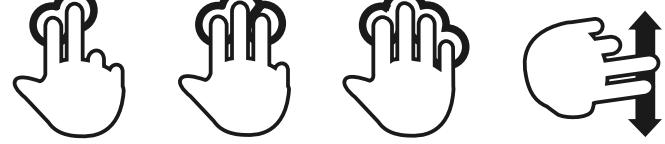
```
func application(application: UIApplication,  
didFinishLaunchingWithOptions ...
```

```
func applicationWillResignActive(application:  
UIApplication) {}
```

```
func applicationDidEnterBackground(application:  
UIApplication) {}
```

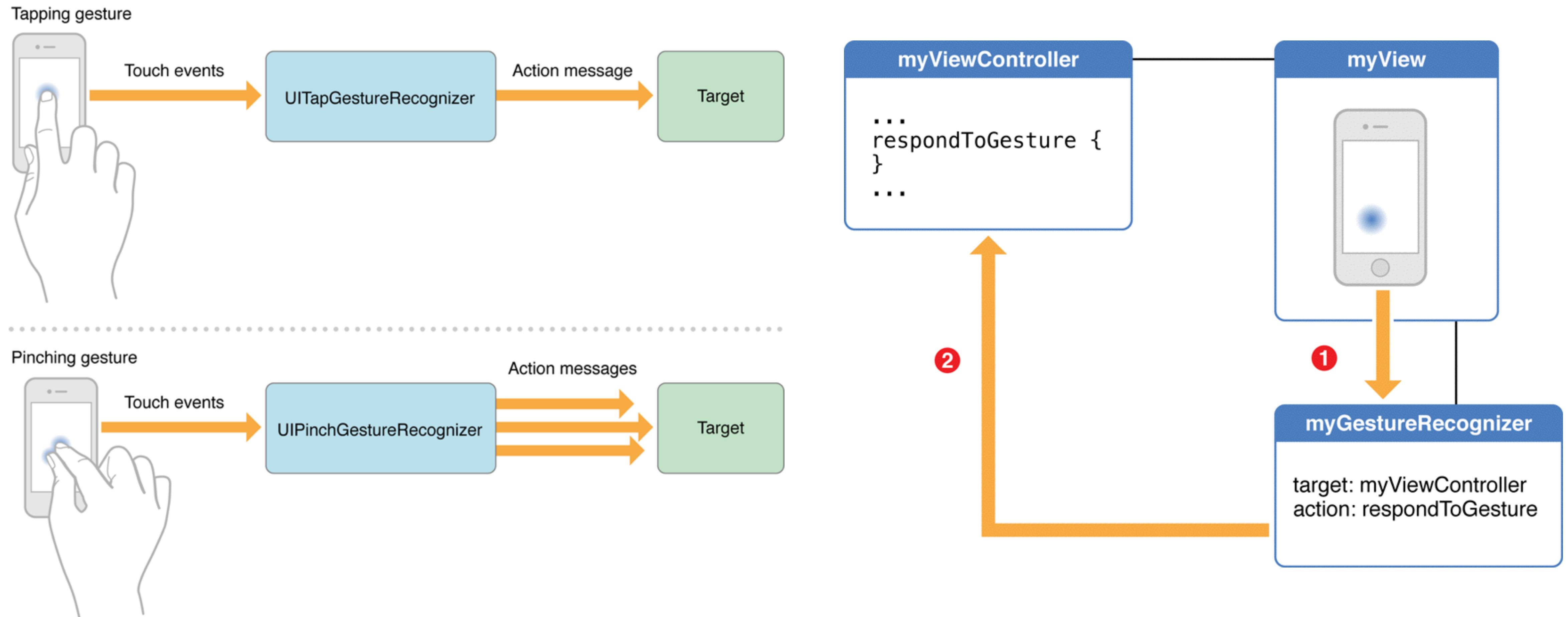


# Standard Gestures

View	Purpose
 Tap	To press or select a control (or item)
 Double tap	To zoom in/out and center a block of content or an image
 Touch & hold	In editable text, to change the cursor position
 Swipe/Pan	To scroll or pan (In a table-view row, to reveal the additional buttons)
 Flick	To Scroll or pan quickly
 Pinch open	Zoom in
 Pinch close	Zoom out
	many more

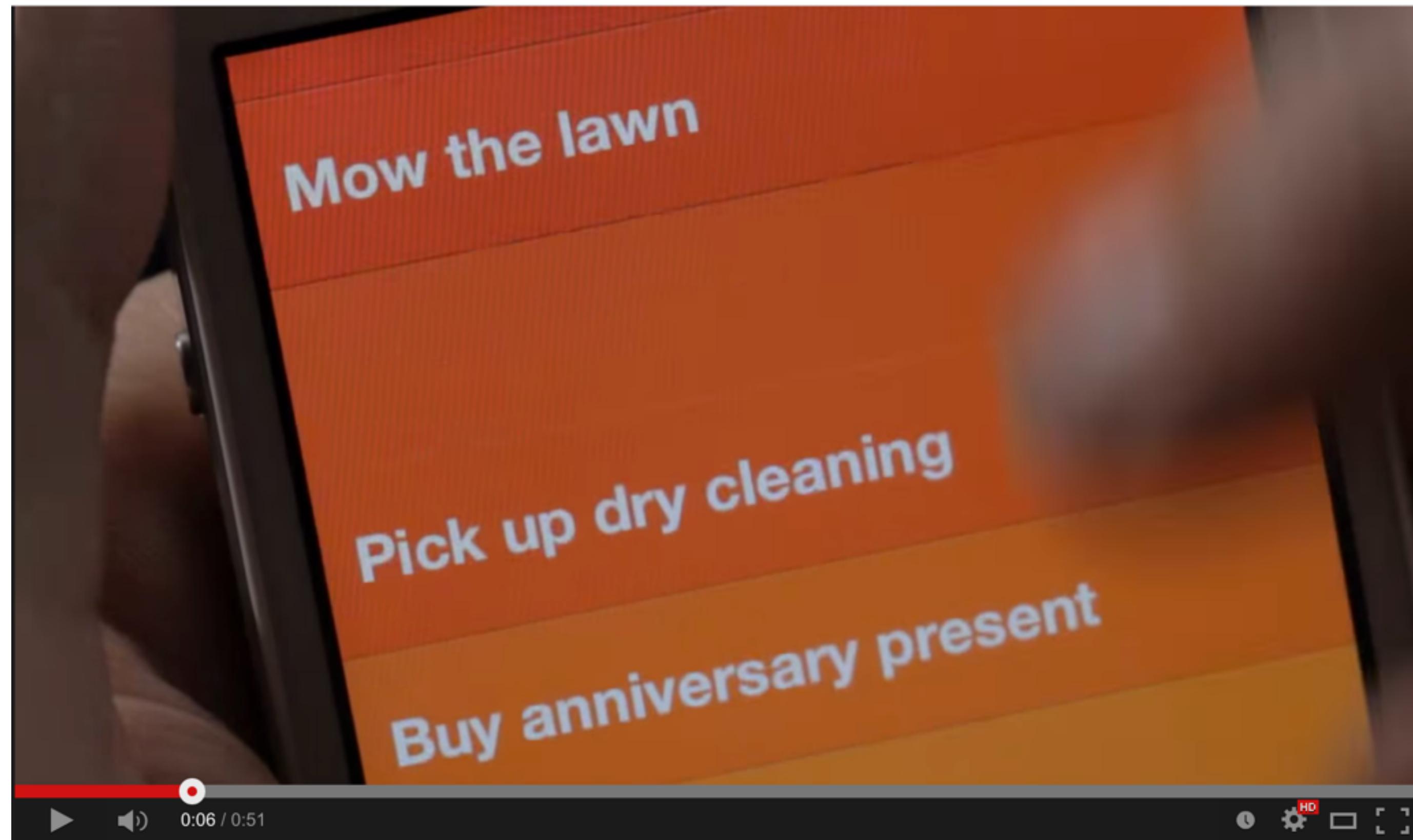
Source: [Apple - Start Developing iOS Apps Today](#)

# Gesture Recognizers



Source: [Event Handling Guide for iOS](#)

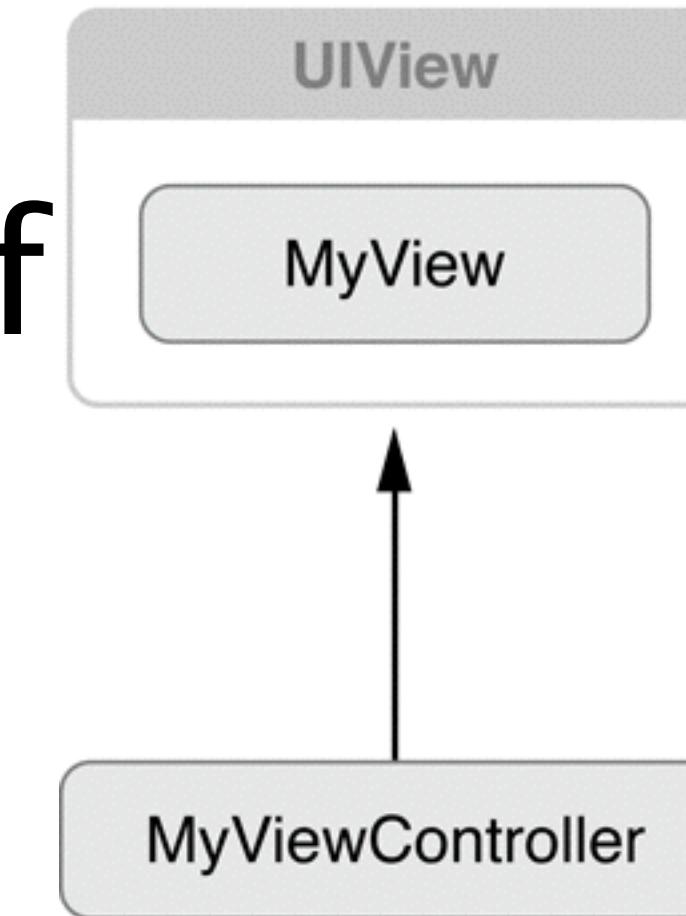
# Clear App



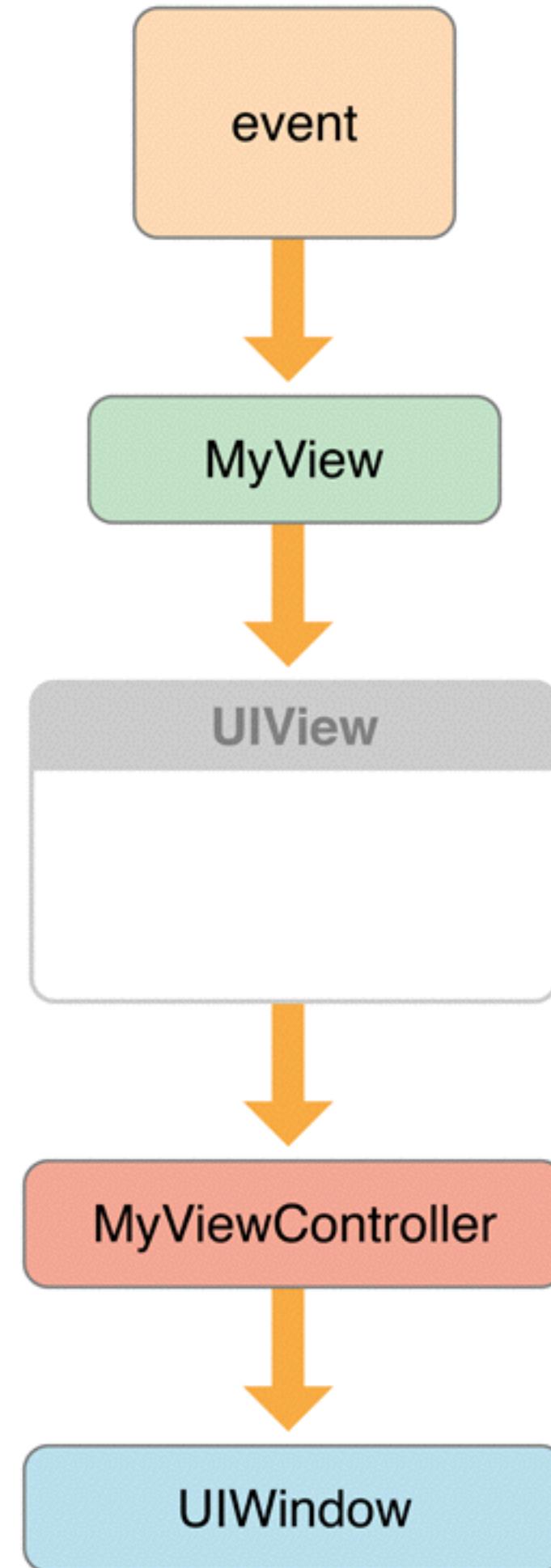
# Responder Chain (1/2)

- Defines How Events Are Propagated to the App
- The responder chain is a series of linked responder objects.
- It starts with the first responder and ends with the application object.
- If the first responder cannot handle an event, it forwards the event to the next responder in the responder chain.

View object hierarchy



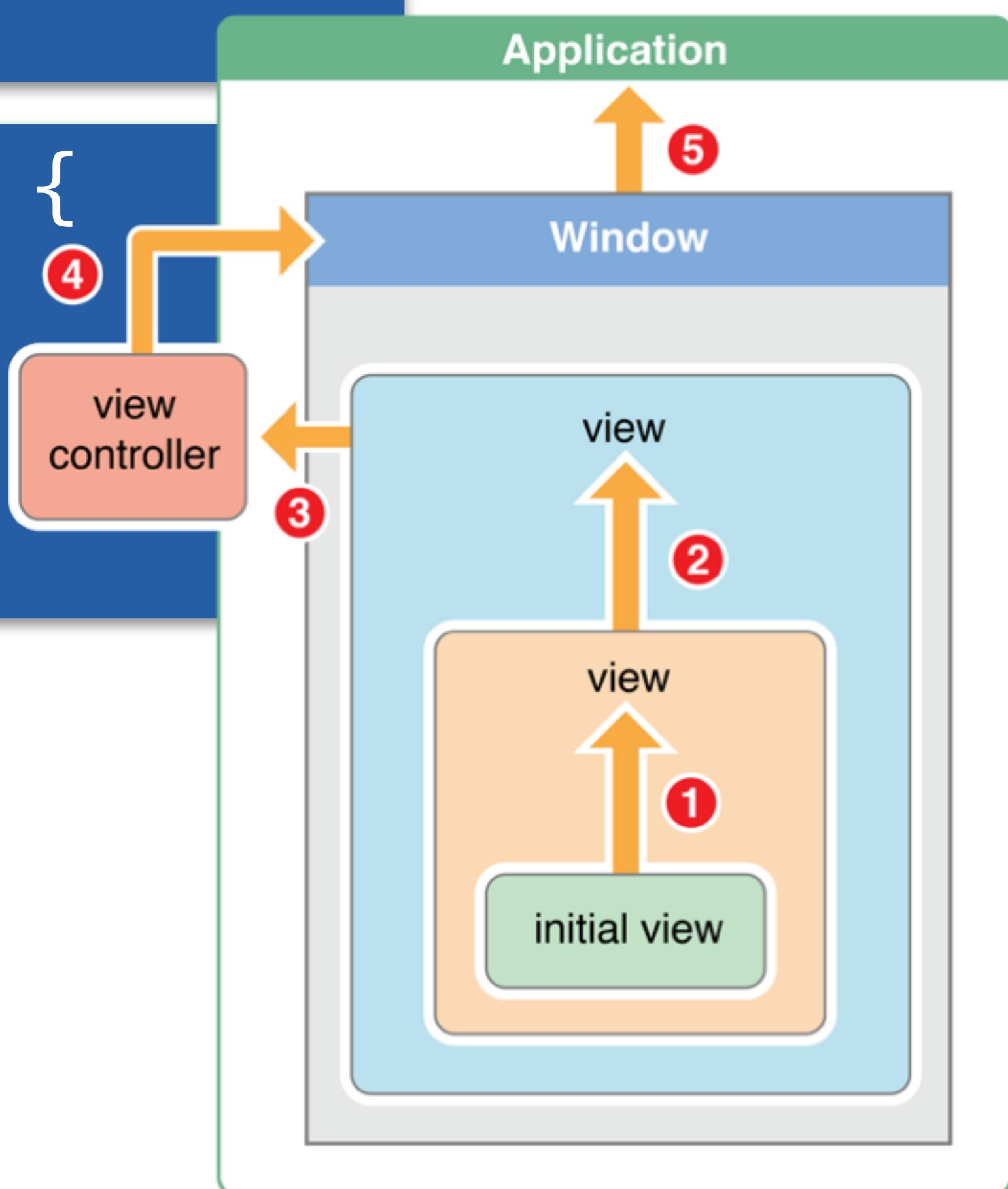
Event flow



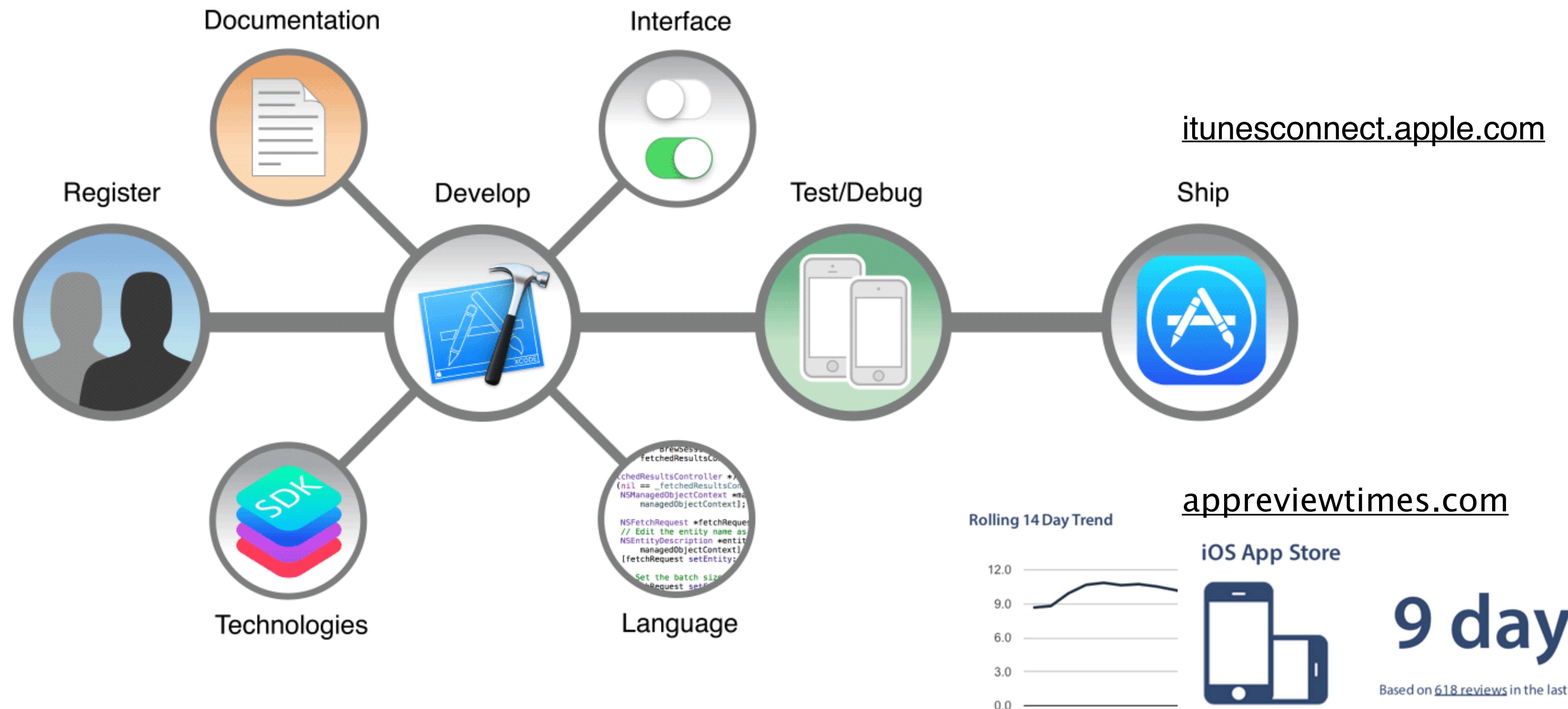
# Responder Chain (2/2)

```
override func canBecomeFirstResponder() -> Bool {  
    return true  
}
```

```
override func viewDidAppear(animated: Bool) {  
    super.viewDidAppear(animated)  
    self.becomeFirstResponder()  
}
```



# Overview



Source: Apple - Start Developing iOS Apps Today